Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Звіт з лабораторних робіт

з дисципліни

«Моделювання програмного забезпечення»

Виконав:

студент 3-го курсу

групи ІП-53, ФІОТ

Сулима Олександр

Сергійович

# Київ 2017

# Лабораторна робота №7

Створення діаграми розміщення

Мета роботи: отримати навички побудови діаграми розміщення.

Завдання:

1. Побудувати діаграму розміщення для обраної автоматизованої системи.
   1. Ідентифікація всіх апаратних, механічних та інших типів пристроїв, що необхідні для виконання системою своїх функцій.
   2. Подальша побудова діаграми розміщення пов’язана з розміщенням всіх виконавчих компонентів діаграми на вузлах системи.
2. Створити короткий опис діаграми.

Зміст звіту:

1. Діаграма розміщення.
2. Опис вузлів та компонентів.

Виконання роботи

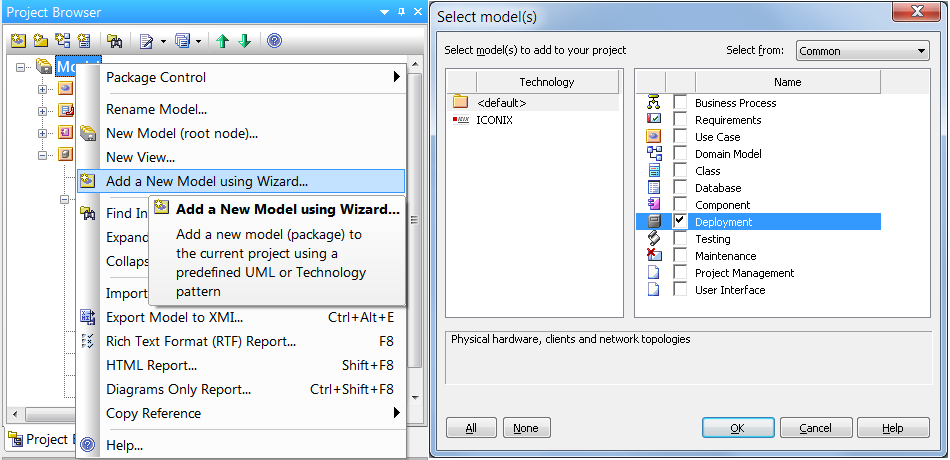
## Побудова діаграми розміщення

Спочатку потрібно визначити компоненти системи. Будувати діаграму розміщення для всієї автоматизованої системи буде не зручно. Тому розглянемо підсистему демонстрації каталогу продукції.

Перерахуємо компоненти системи:

1. Персональний комп’ютер менеджера по роботі з клієнтами.
   * Компонент додатку, що виводить каталог на екран демонстраційного дисплею.
   * Бібліотека, що містить функції доступу до вмісту бази даних.
   * Бібліотека, що містить функції друку обраного елемента каталогу
2. Демонстраційний дисплей.
3. Принтер.
4. Сервер бази даних.
   * SQL база даних, що містить дані, необхідні для створення каталогу.
5. Локальна мережа

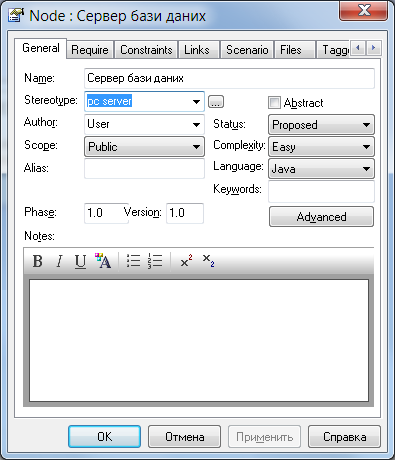
Для додавання моделі розміщення, потрібно вихвати контекстне меню корінного каталогу проекту та обрати пункт «Add a New Model Using Wizard». У вікні, що з’явиться, поставте позначку напроти «Deplyment» та натисніть кнопку «ОК».



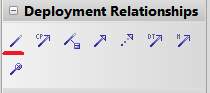
Для побудови діаграми знадобляться наступні елементи панелі «Toolbox»:

Являє собою деякий фізично існуючий елемент системи, що володіє деяким обчислювальним ресурсом. Також може відображати механічні та електричні пристрої: сенсори, принтери, сканери, маніпулятори тощо.

Щоб задати тип елементу, у контекстному меню елемента оберіть пункт «properties», та у полі «stereotype» вкажіть необхідний тип, або введіть його власноруч з клавіатури.



Складова частина вузла. Може бути як фізичним пристроєм, так і програмним додатком, бібліотекою тощо. Тип задається аналогічно до типу вузла. Щоб додати компонент до складу певного вузла достатньо просто перетягнути його мишкою з робочого поля діаграми на потрібний вузол.

Всі з’єднання виконуються стрілкою «Associate».

Готова діаграма:



## Опис елементів діаграми розміщення.

Мета даної моделі розміщення – визначити фізичне положення компонентів розподільної системи, що буде забезпечувати виконання всіх функцій підсистеми в тих місцях, де користувачам буде найбільш зручно.

Дана підсистема має наступні вузли:

* *Сервер бази даних* – комп’ютер, що призначений для зберігання бази даних. Має наступні компоненти:
  + *SQL база даних* – містить списки меблевих проектів, наявних матеріалів та інструментів, щоденних робочих планів.
* *ПК менеджера* – комп’ютер менеджера по роботі з клієнтами. Головне призначення якого – демонстрація клієнтові меблів та супровід замовлення. Має наступні компоненти:
  + *Show.exe* – компонент додатку, що призначений для керування відображенням елементів каталог у на демонстраційному дисплеї.
  + *DataToManagement* – бібліотека, що містить функції отримання даних з бази даних.
  + *Print* – бібліотека, що містить функції формування та друку даних заданих елементів каталогу.
* *Принтер* – електронний прилад для друку документів.
* *Демонстраційний дисплей* – ЖК-дисплей високої роздільної здатності, що слугує для демонстрації продукції підприємства.
* *Локальна мережа* – фізичне об’єднання комп’ютерів на території підприємства.

**Питання:**

1. Напишіть основну характеристику діаграми розміщення

Діаграма в UML, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах.[1] Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

1. Які існують взаємодії в діаграмі розміщенні

Діаграма розміщення відображає фізичні взаємозв'язку між програмними і апаратними компонентами системи. Вона є хорошим засобом для того, щоб показати маршрути переміщення об'єктів і компонентів в розподіленої системі. Кожен вузол на діаграмі розміщення являє собою певний тип обчислювального пристрою, в більшості випадків - частина апаратури. Компоненти на діаграмі розміщення становлять фізичні модулі програмного коду. Як правило, вони в точності відповідають компонентам на діаграмі компонентів. Таким чином, діаграма розміщення відображає виконання кожного компонента в системі. На практиці діаграми розміщення використовуються не дуже часто. Багато розробники дійсно користуються такими діаграмами, проте вони являють собою просто неформальні картинки. З іншого боку, кожна система має свої власні фізичні характеристики, які бажано явно виділити, і в подальшому буде потрібно велика ступінь формалізму.

1. Пакети в діаграмі розміщення

Об’єднання елементів у пакети (так зване пакетування) може відбуватися з різних міркувань, наприклад, якщо вони використовуються сумісно або створені одним автором, або стосуються певного аспекту розгляду, як-от інтерфейс з користувачем, пристрої введення/ виведення і под. На стадії реалізації до одного пакета може бути віднесено всі підсистеми, що в діаграмі розміщення (розглянуто вище) віднесено до одного вузла. Призначення пакета - бути елементом конфігурації, тобто елементом, який можна включати як визначену складову композиції в побудові певної системи. На пакет можна посилатися у різних діаграмах, котрі можуть розробляти окремі команди спеціалістів. Терміном конфігурація (configuration) будемо позначати отримання програмної системи шляхом добору окремих екземплярів модулів з визначеного наперед складу їхніх варіантів.

1. Стадія проетування розміщення

Стадія системного проектування містить у собі рішення верхнього рівня щодо розробки системи в цілому. Тут здійснюється розробка архітектури системи за допомогою діаграми розгортання. Вона дозволяє виділити обчислювальні ресурси, пристрої, що використовуються ними, і з'єднання між ними, спроектувати розміщення на визначених пристроях окремих процесів і компонентів, що виконуються, що особливо важливо при проектуванні складних систем і інтернет-додатків.

1. Конфігурація системи

Операційна система може мати у своєму складі конфігурацію модулів, що дозволяють взаємодію з різними пристроями, але лише окремі з них приєднані до даного комп’ютера, для якого створюється версія операційної системи як конкретна конфігурація з визначеної множини; система керування польотом літака має у своєму складі конфігурацію модулів, що забезпечують введення показників приладів конкретного борту літака. Пакет часто може передбачати кілька версій конфігурації його складових.

Національний технічний університет України

«Київський політехнічний інститут»

Факультет інформатики та обчислювальної техніки

Звіт з лабораторних робіт

з дисципліни

«Моделювання програмного забезпечення»

Виконав:

студент 3-го курсу

групи ІП-53, ФІОТ

Сулима Олександр

Сергійович

# Київ 2017

# Лабораторна робота №8

Зворотне проектування

**Мета роботи:** Ознайомитись з принципами зворотного проектування. Навчитися генерувати діаграми класів з початкового коду.

**Завдання:**

1. Використати діаграму класів одного з процесів з попередньої лабораторної роботи та написати для неї програмний код. В коді програми обов’язково реалізувати всі види зв’язку, та описати всі поля і методи, що були позначені на діаграмі класів.
2. Використовуючи програмний засіб для побудови UML-діаграм, на основі написаного коду згенерувати UML-діаграму класів.
3. Порівняти початкову UML-діаграму, з попередньої лабораторної роботи з отриманою. В випадку якщо діаграми відрізняються описати в чому різниця та чим вона пояснюється.

Зміст звіту:

1. Діаграма класів, на основі якої буде написано програмний код.
2. Програмний код відповідно діаграми класів з пункту 1.
3. Побудована діаграма класів методом оберненого проектування, на основі коду, описаного в пункті 2.
4. Порівняння діаграми класів з пункту 1і пункту 3. Пояснення різниці(якщо вона є).

Виконання роботи

## Попередня діаграма класів

Для виконання лабораторної роботи програний код будемо писати на мові C++ в середовищі Microsoft Visual Studio, що також має інструменти для генерації діаграми класів.

Напишемо код, для діаграми класів, що була створена в лабораторній роботі №2, додавши деякі логічні елементи, що не були враховані раніше.



Як видно з діаграми, на ній присутні абстрактний клас, відношення наслідування, відношення реалізації та відношення агрегації. Це основні об’єкти та відношення діаграми класів, тому буде правильним перевірити правильність їх відображення при оберненому проектуванні.

## Програмний код

Item

class Item{

//variables

public:

string ID;

string ID\_Delivery;

string Name;

string Type;

string Manufacturer;

string State;

unsigned int IssueDate;

string IssueReceiver;

//functions

public:

virtual string GetDescription() const = 0;

void SetIssued(unsigned int date, string &Receiver); //Встановлення флагу видачі (дата, отримувач)

};

Stuff

class Stuff : public Item{ //матеріал, напівфабрикат, деталь тощо

//variables

public:

float height;

float width;

float length;

string Material;

//functions

public:

virtual string GetDescription() const;

};

ElectricTool

class ElectricTool : public Item{ //електроінструменти

//variables

public:

unsigned int Voltage;

unsigned int Power;

string Battery;

unsigned int BatteryCapacity;

//functions

public:

virtual string GetDescription() const;

};

Stock

class Stock{

//variables

public:

list<Item\*> Items;

//functions

public:

void IssueItem(string &ID\_item, unsigned int date, string Name);

void ShowItem(string &ID\_item);

void AddNewItem(Item &Unit);

private:

Item\* Find(string &ID\_item);

};

Supplier

class Supplier{

//variables

public:

const string ID;

string Name;

string Address;

string Phone;

list <Delivery\*> Deliveries;

//constructors

public:

Supplier(string Data[4]);

//functions

public:

void AddDelivery(string &ID, unsigned int date, list<Item\*> &Items);

};

Delivery

class Delivery{

//variables

public:

const string ID;

Supplier\* Sup;

unsigned int date;

list<Item\*> Items;

//constructors

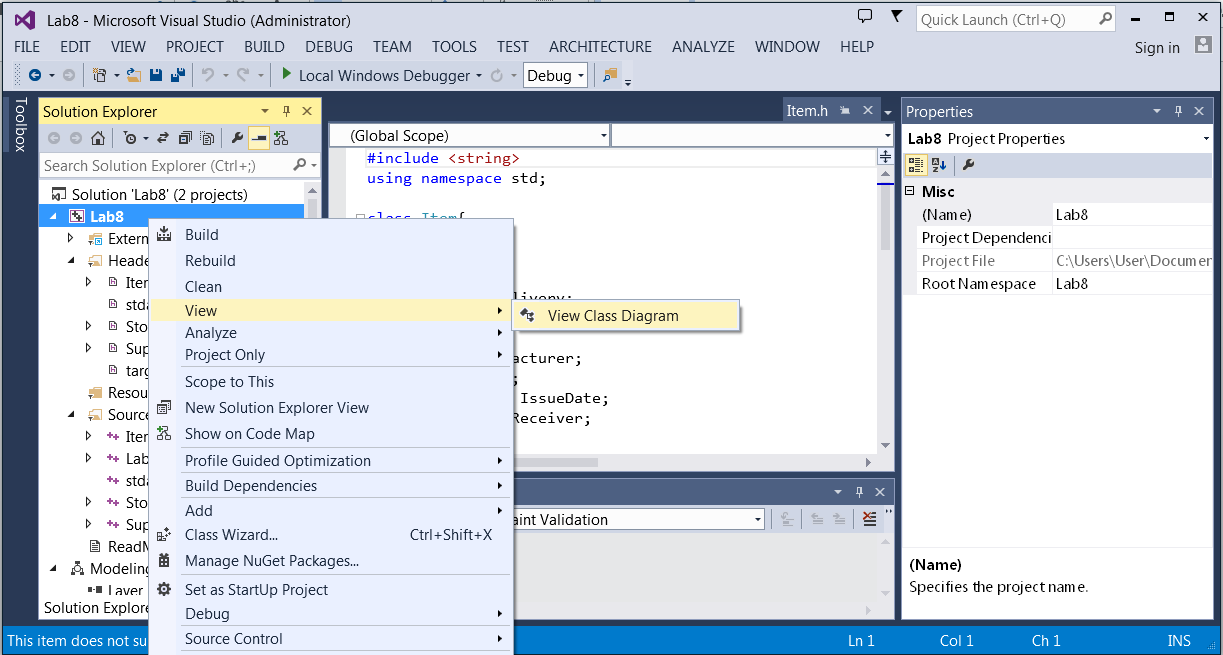
public:

Delivery(string &ID, unsigned int date, list<Item\*> &Items, Supplier\* Sup);

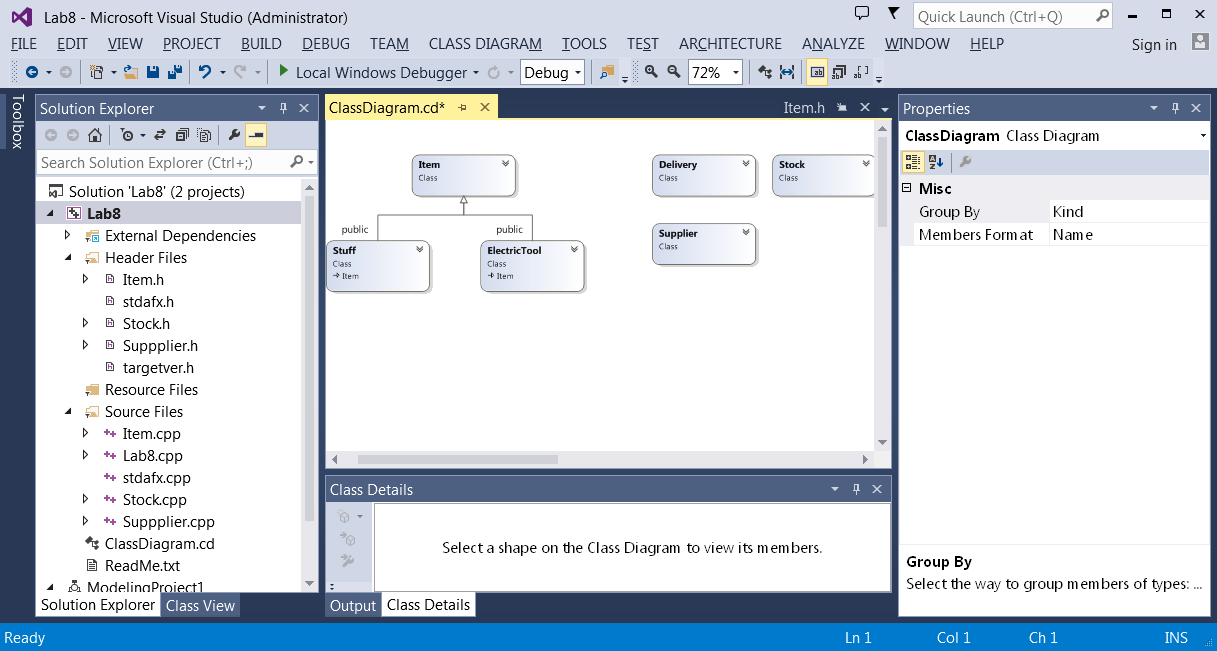
};

## Побудова діаграми класів методом оберненого проектування

Для побудови діаграми класів скористаємось можливостями Microsoft Visual Studio. Для цього на панелі Solution Explorer оберіть потрібний проект, та у контекстному меню проекту оберіть пункт View – View Class Diagram.

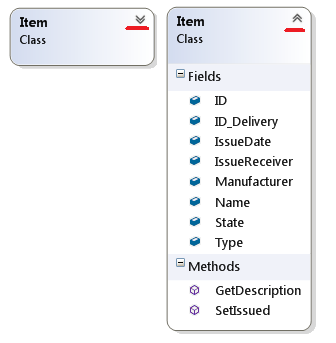


Microsoft Visual Studio автоматично згенерує діаграму для всіх класів проекту.



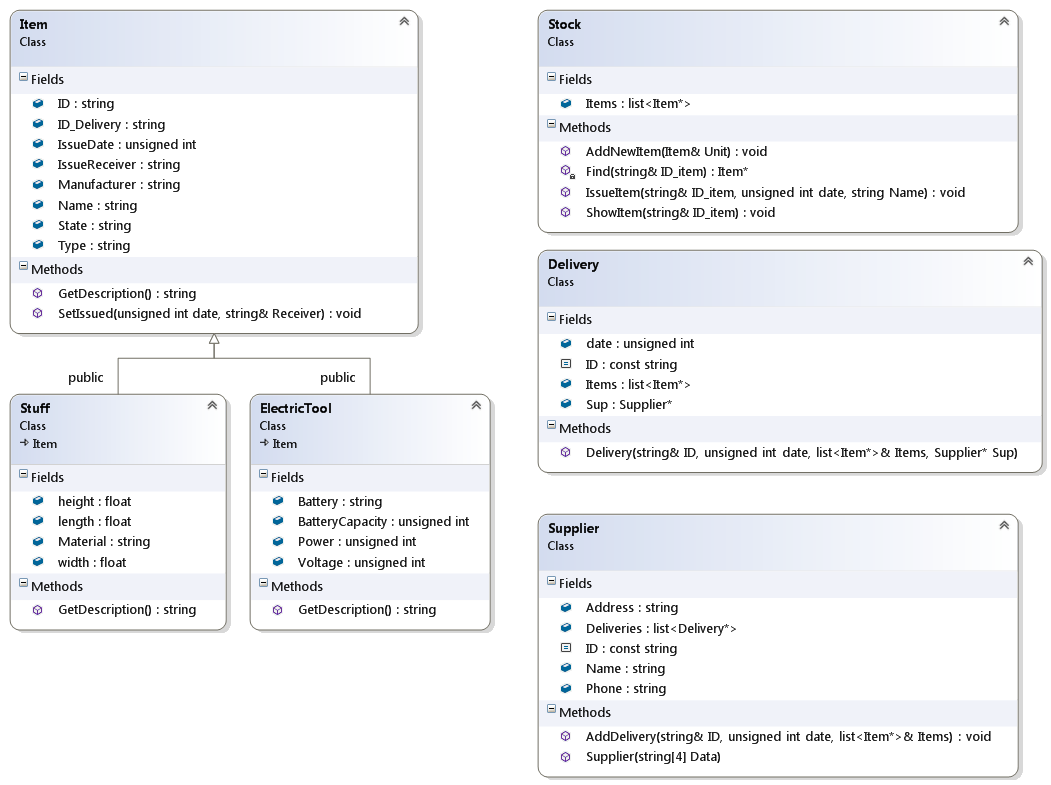
Масштаб зображення можна змінити скориставшись головним меню додатку: Class Diagram – Zoom.

Щоб відобразити поля та методи класів, слід натиснути на стрілку у кутку класу на діаграмі.

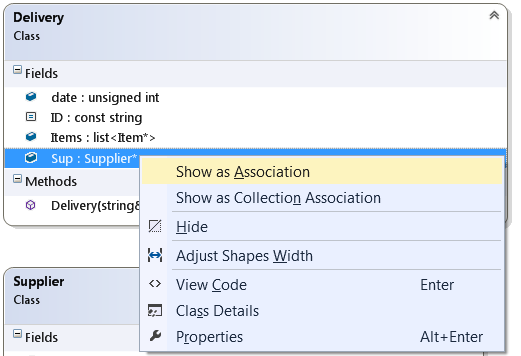


Для того, що відобразити повну інформацію щодо полів та методів слід натиснути правою кнопку миші по робочому полі діаграми та обрати наступний пункт меню: Change Member Format – Display Full Signature.

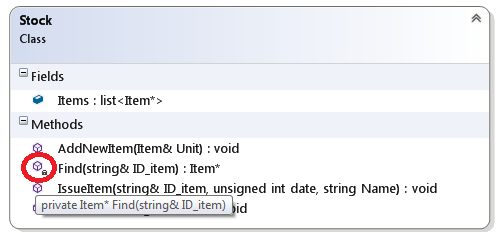
Після цих кроків діаграми прийме наступний вигляд:



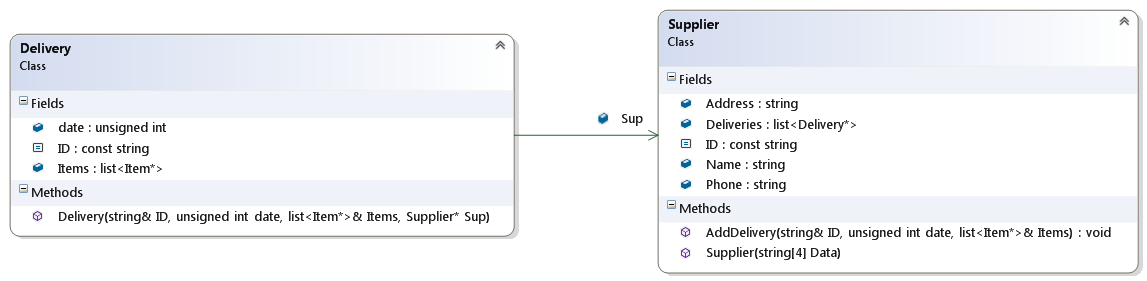
Діаграмі досі не вистачає відношень асоціації та агрегації. Для того, що відобразити ці відношення, потрібно визвати контекстне меню необхідного поля відповідного класу та обрати пункт Show As Association.



Щоб переконатися в коректності рівнів видимості полів класу, слід навести курсор на позначку видимості та прочитати зміст підказки.

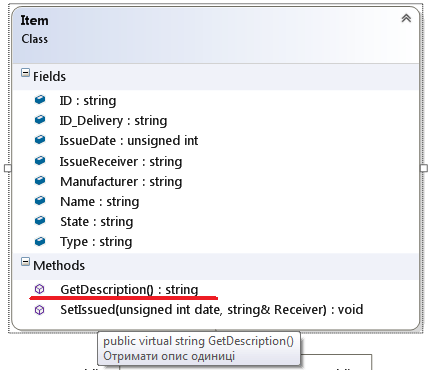


Так виглядає відношення асоціації між класами *Delivery* та *Supplier:* поле Sup виноситься назовні та вказується над стрілкою відношення.

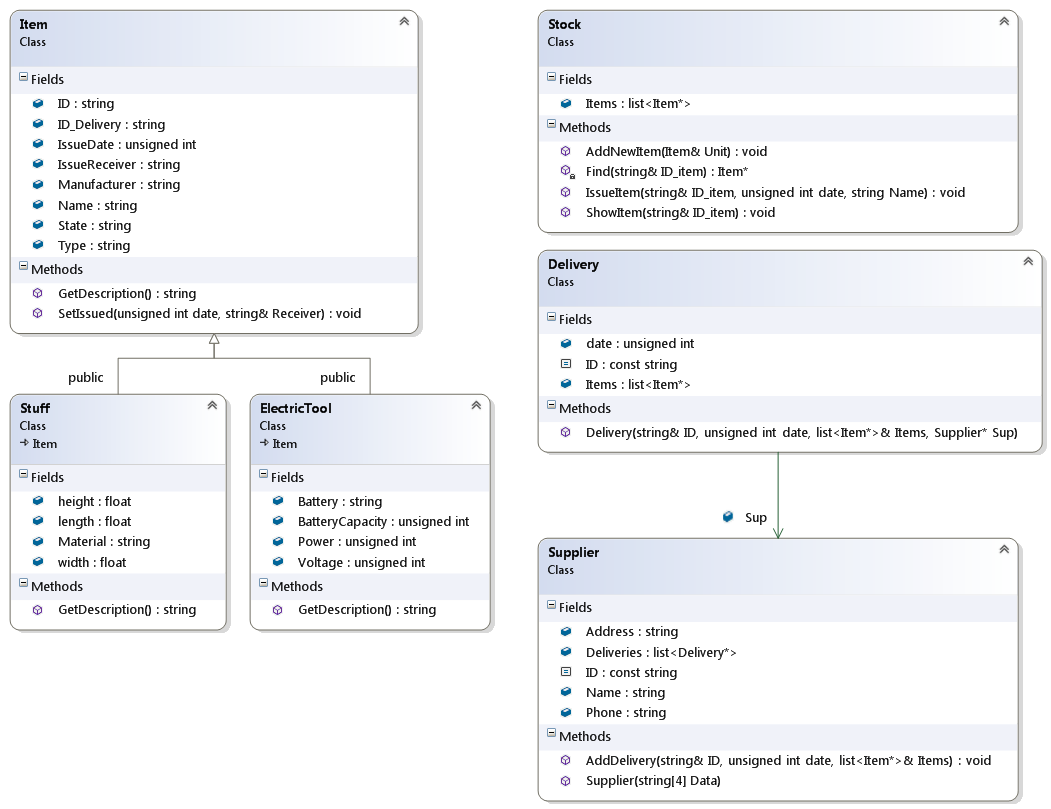


Увага! Visual Studio не відобразить відношення агрегації так, як це зроблено на власноруч побудованій діаграмі. Справа в тому, що клас *Delivery* з точки зору середовища містить не клас *Item*, а клас *list*, який вже в свою чергу містить масив вказівників на клас *Item*.

Також графічно ніяк не позначаються чисто віртуальні функції. У правильності генерації діаграми можливо впевнитись лише переглянувши підказку, що з’явиться при наведенні курсора на поле класу.



Готова згенерована діаграма має наступний вигляд. Крім вищезгаданої відмінності, слід зауважити, що Visual Studio на даному етапі не розрізняє абстрактні та звичайні класи оскільки C++ не має спеціальних слів для позначення абстрактного класу (наприклад, abstract class у випадку Java), тому клас *Item* на діаграмі він відображається як звичайний клас.



**Питання**

1. Дайте оцінку зворотнього проектування

Дослідження деякого пристрою чи програми з метою розуміння принципів роботи досліджуваного об'єкта. Найчастіше використовується з метою створення об'єкта, за функціональністю аналогічного досліджуваному але без точного копіювання його функцій. Зазвичай використовується у випадках, коли розробник оригінального пристрою чи програми не надає точних відомостей про алгоритми функціонування виробу, або будь-яким чином намагається завадити використанню технологій, що задіяні його функціоналом. У наш час зворотна розробка найширше вживається для аналізу програмного забезпечення з закритими вихідним кодом. При цьому людина, що використовує даний метод дослідження, аналізує машинний код програми (зазвичай в дизасембльованому вигляді) та складає алгоритм функціонування ПЗ для подальшої реалізації в окремому продукті або відтворює його у відповідних специфікаціях.

1. Використання зворотнього проектування

Дослідження програм часто використовують з метою подальшої їхньої модифікації, розширення функціоналу або створення засобів, що дозволяють обходити обмеження на їхнє використання (наприклад умовно-безкоштовні програми). Також такі методи можуть використовуватися для отримання специфікації протоколів обміну інформацією (наприклад мережевих протоколів) або способів її зберігання (формати файлів).

1. Методи

Моніторинг активності. Таким способом найчастіше проводиться дослідження протоколів обміну інформацією. Наприклад для дослідження мережевих проколів може використовуватися перехоплення потоків даних в мережі за допомогою спеціалізованих програмних чи/та апаратних засобів. Цей метод може не дати повного уявлення про алгоритми функціонування ПЗ. Дизасемблювання. Машинний код програми читається та перекладається мовою асемблера для свого розуміння в чистому вигляді. Таким способом можна досліджувати будь-яке програмне забезпечення, але за допомогою використання певних технологій при розробці ПЗ дизасемблювання можна значно ускладнити. Метод вимагає високої кваліфікації людини, що проводить зворотну розробку та великих затрат часу. Декомпіляція. Полягає у перекладі машинного коду програми мовою високого рівня. Метод важко реалізувати з огляду на складність створення інструментів.

1. Групи проектування

Група Reverse Engineer Fromдозволяє задати джерело зворотного проектування - базу даних або SQL (DDL) -скріпт. За допомогою кнопки Browse можна вибрати текстовий файл, що містить SQL-скрипт.

1. Реверсивна група

Група Items to Reverse Engineerдозволяє задати об'єкти бази дан них, на основі яких буде створена модель. За допомогою списку вибору Option Set, а також кнопок New, Update і Delete можна створювати і редакті- ровать іменовані конфігурації об'єктів бази даних, які можуть бути використані багаторазово при інших сеансах зворотного проектування.