Метою роботи: є вдосконалення автоматизованої інформаційної системи за допомогою CASE - засобів візуального моделювання та дослідження її можливостей за допомогою Rational Rose, що є сучасним і потужнім засобом аналізу, моделювання і розробки програмних систем.

ОСНОВНІ ЗАДАЧІ ЩО РЕАЛІЮЮТЬСЯ В РОБОТІ:

- розробити базу даних та проаналізувати її структуру;
- дослідити інформаційну систему за допомогою візуального представлення елементів та залежностей моделювання CASE-засобами уніфікованої мови UML;
- розробити зручний графічний інтерфейс користувача клієнтських застосувань за допомогою програмного забезпечення;
- дослідити програмне забезпечення способами тестування необхідними для підвищення рівня безпеки.

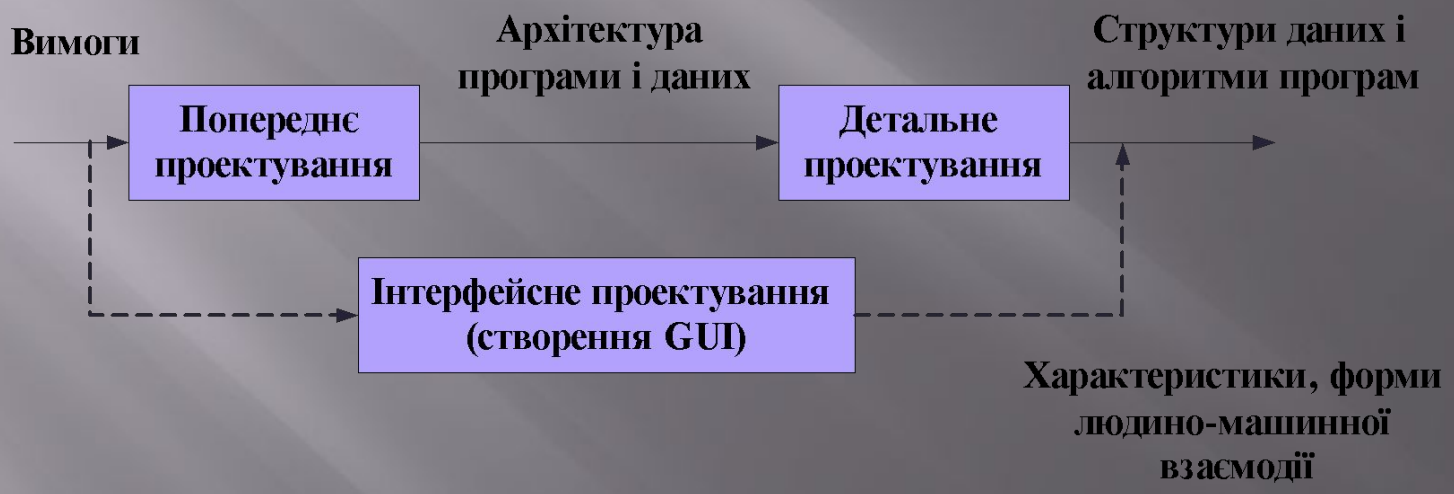
ПРОЕКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

Технологією проектування вибрана *спіральна модель*, яка базується на кращих властивостях класичного життєвого циклу та макетування, до яких додається новий елемент - аналіз ризику.

1 - початковий збір вимог і планування проекту;
2 - та ж робота, але на основі рекомендацій замовника;
3 - аналіз ризику на основі початкових вимог;
4 - аналіз ризику на основі реакції замовника;
5 - перехід до комплексної системи;
6 - початковий макет системи;
7 - наступний рівень макета;
8 - сконструйована система;
9 - оцінювання замовником.

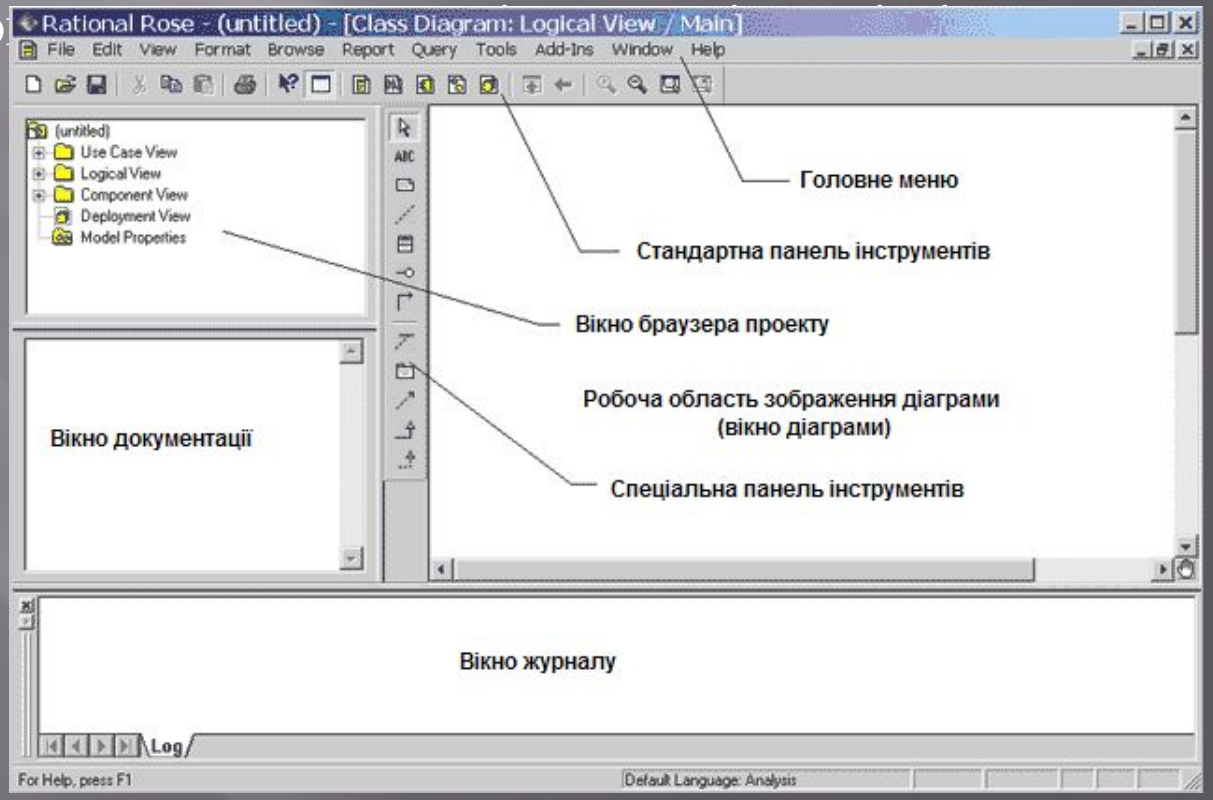
ПРОЕКТВАННЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА

Проектування - ітераційний процес, за допомогою якого вимоги до програмних систем транслюються в інженерні представлення програмної системи. Зображено інформаційні зв'язки процесу проектування:



ВІЗУАЛЬНЕ МОДЕЛЮВАННЯ

За допомогою Rational Rose і будемо моделювати систему. Rational Rose - це CASE-система для візуального моделювання об'єктно-орієнтованих програмних продуктів. Візуальне моделювання - процес графічного описания розробленого програмного забезпечення. Кнопки панелі інструментів дозволяють вико



вигляд робочого інтерфейсу CASE-засобу IBM Rational Rose

ПРОЕКТУВАННЯ CASE- МОДЕЛІ

CASE-засіб IBM Rational Rose з часу своєї появи зазнав серйозних змін, і в даний час є сучасним інтегрованим інструментарієм для проектування архітектури, аналізу, моделювання і розробки програмних систем. Саме в IBM Rational Rose мова UML стала базовою технологією візуалізації і розробки програмних систем.

Актор (Actor або суб'єкт) – хтось або щось (людина, машина, і т.д.), що не є частиною системи, але взаємодіє з нею.

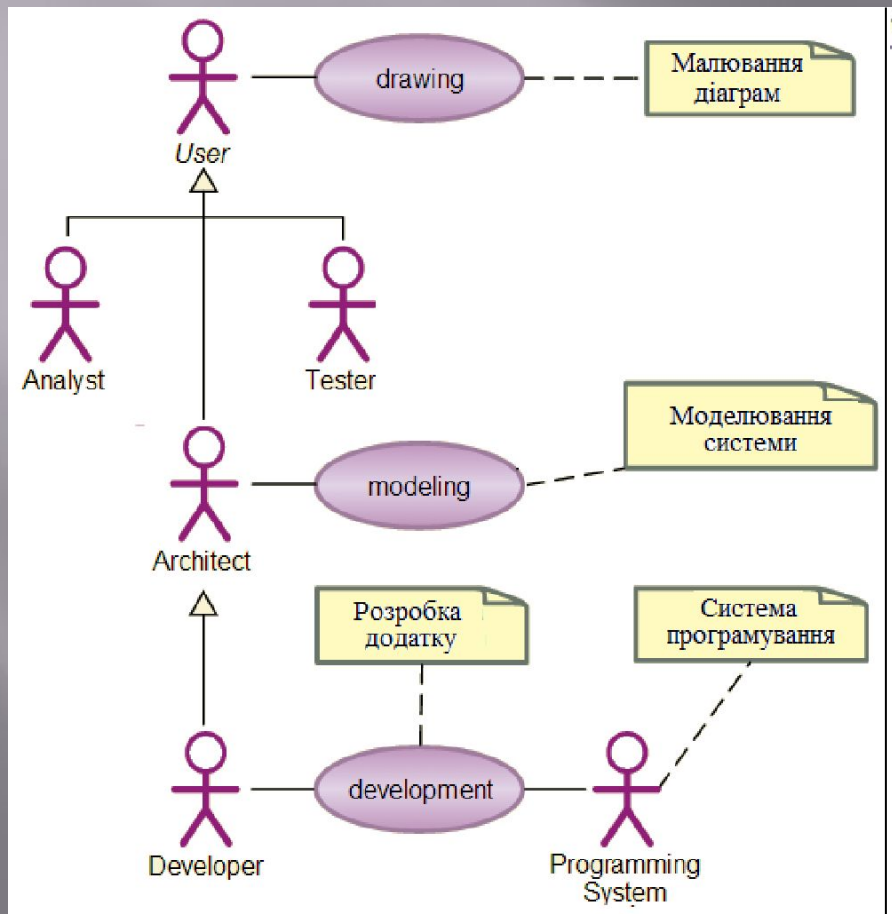
Прецедент (Use Case) – це фрагмент функціональності, яку забезпечує система. Інакше кажучи, прецедент показує, як можна використовувати систему.

Суб'єкти системи «Магазин»

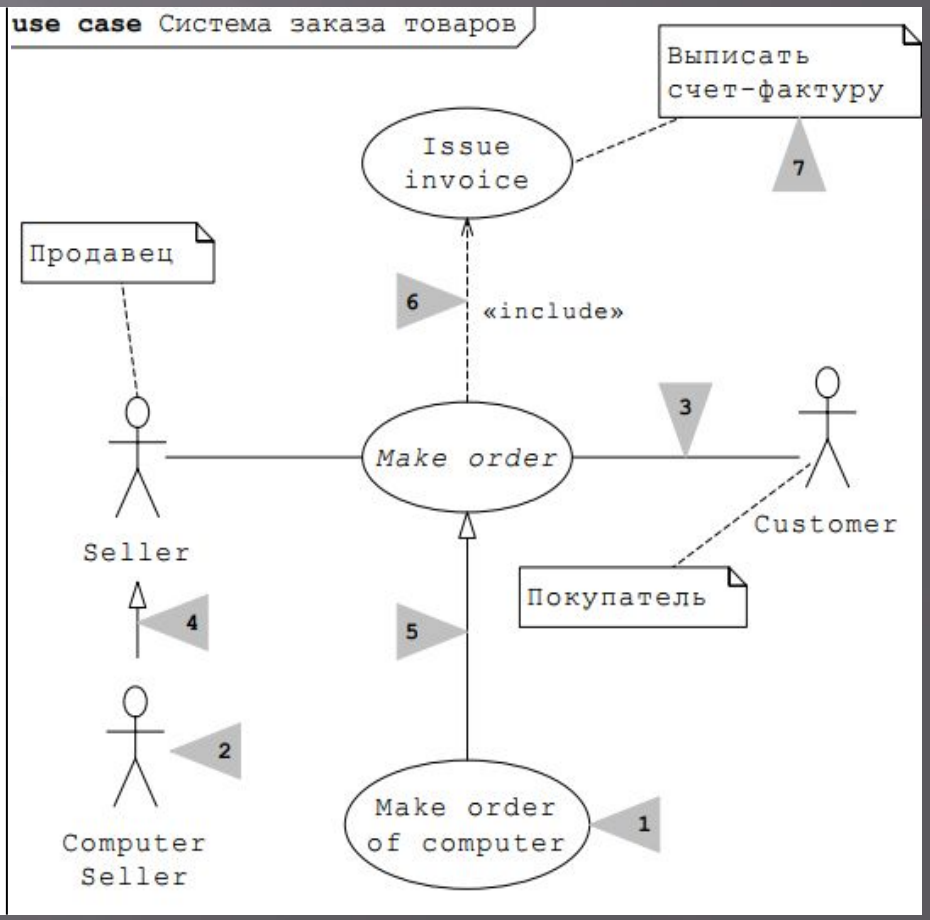
Прецеденти модельованої системи «Склад продукції»

ІНСТРУМЕНТАЛЬНА ПІДТРИМКА

Розглянемо головні аспекти інструментальної підтримки розробки за допомогою однієї з діаграм UML.



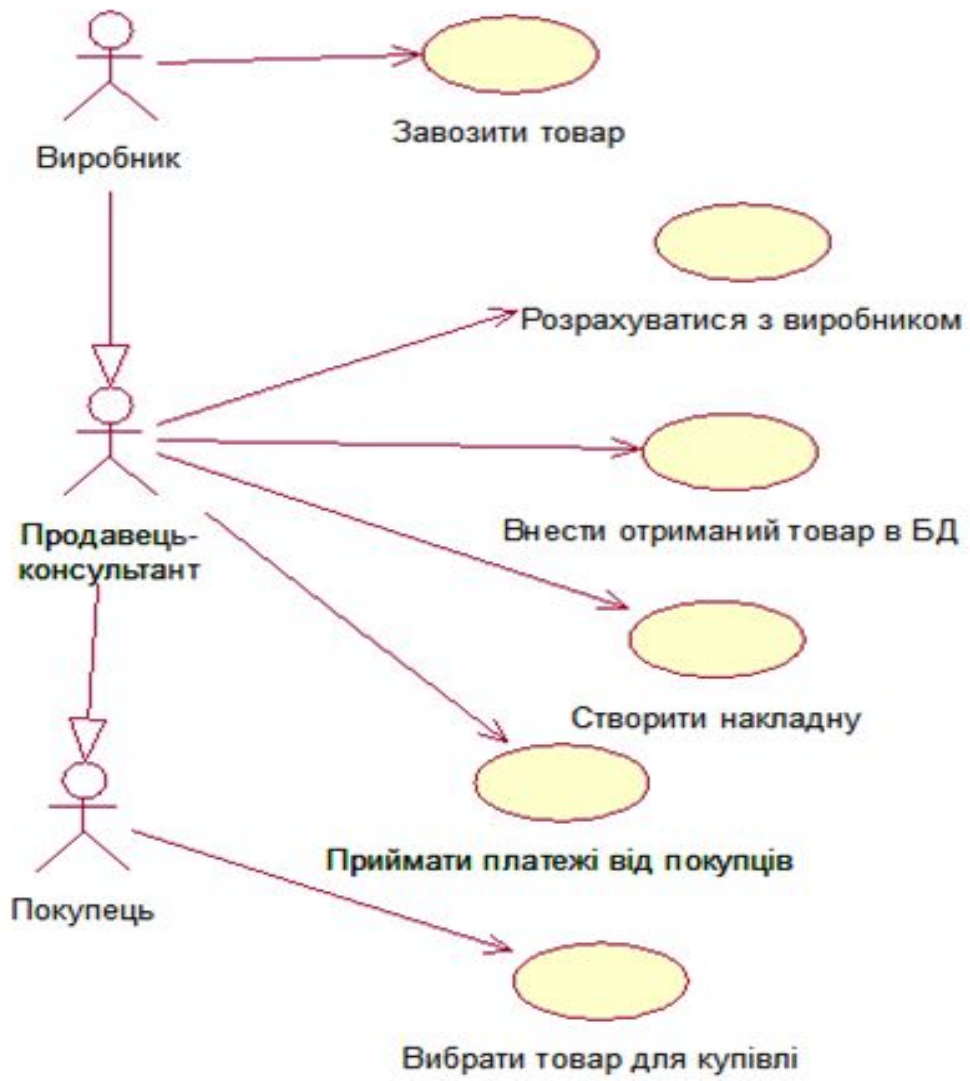
Діаграма використання



Нотація діаграми використання

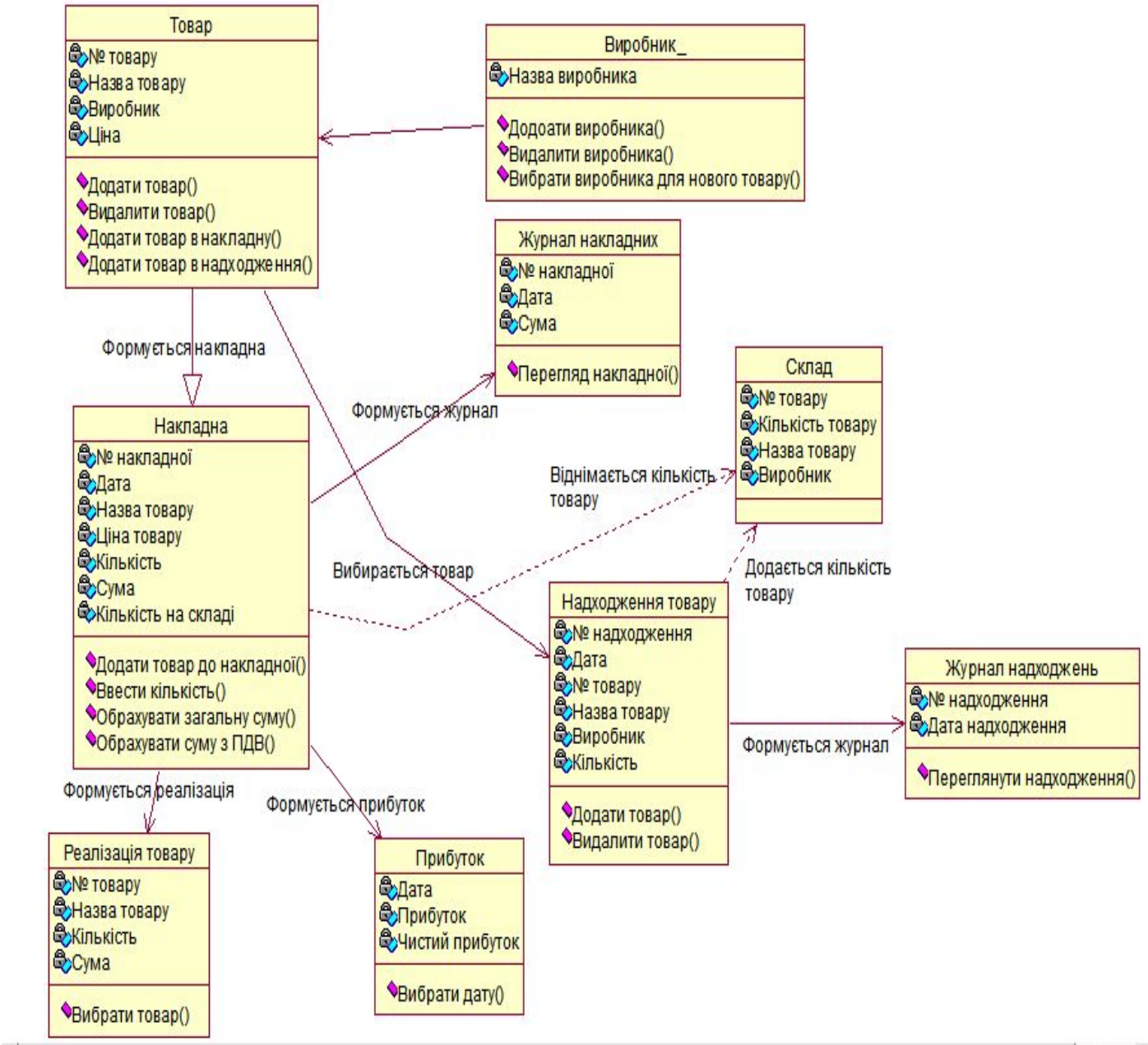
ВИЗНАЧЕННЯ БІЗНЕС-ПРОЦЕСІВ І ПОТОКІВ ДАНИХ

Діаграма прецедентів – це наочне графічне представлення суб'єктів і прецедентів, а також їх взаємодії в системі разом з будь-якими додатковими визначеннями і специфікаціями. Не проста схема, але є повністю документованою моделлю передбачуваної поведінки системи. Зображена діаграма прецедентів (Use Case) для системи «Магазин».



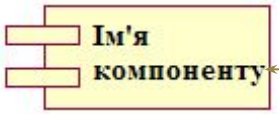
ДІАГРАМА КЛАСІВ

Визначення внутрішнього стану системи показується в моделі класів (class model). Клас (class) - це опис групи об'єктів із загальними властивостями (атрибутами), поведінкою (операціями), відносинами з іншими об'єктами і семантикою. Таким чином, клас є шаблоном для створення об'єкту. Зображена діаграма класів для системи «Магазин».

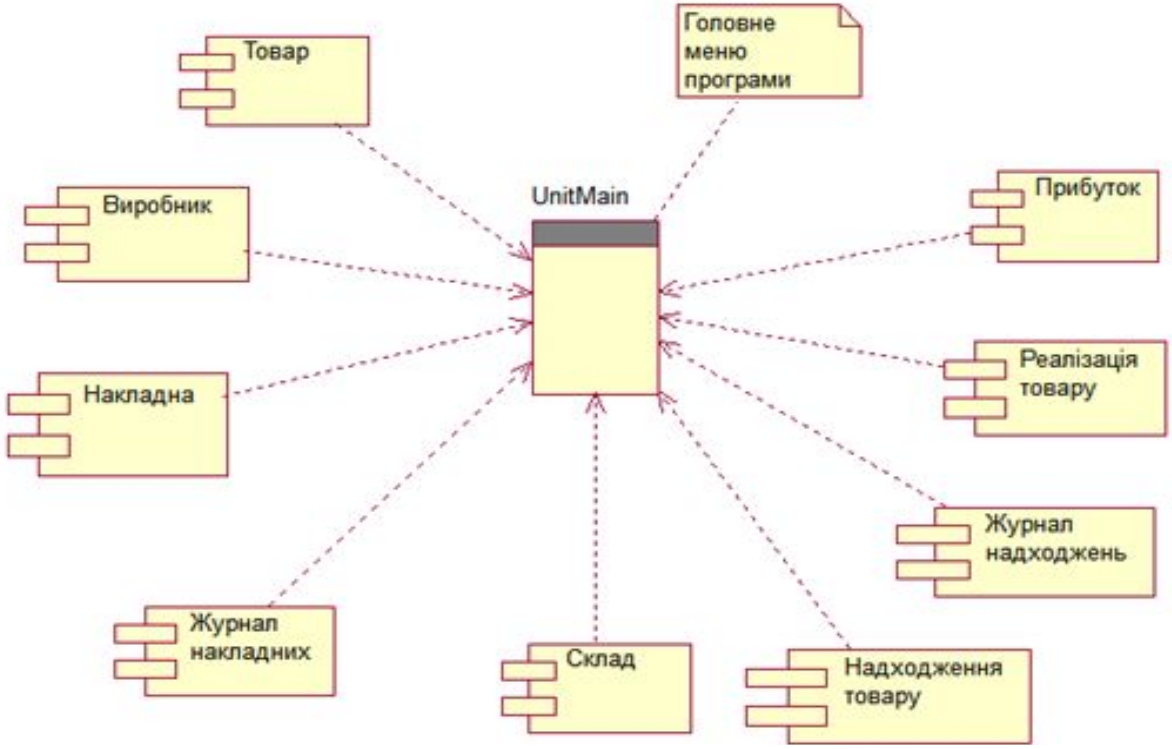


ДІАГРАМА КОМПОНЕНТІВ

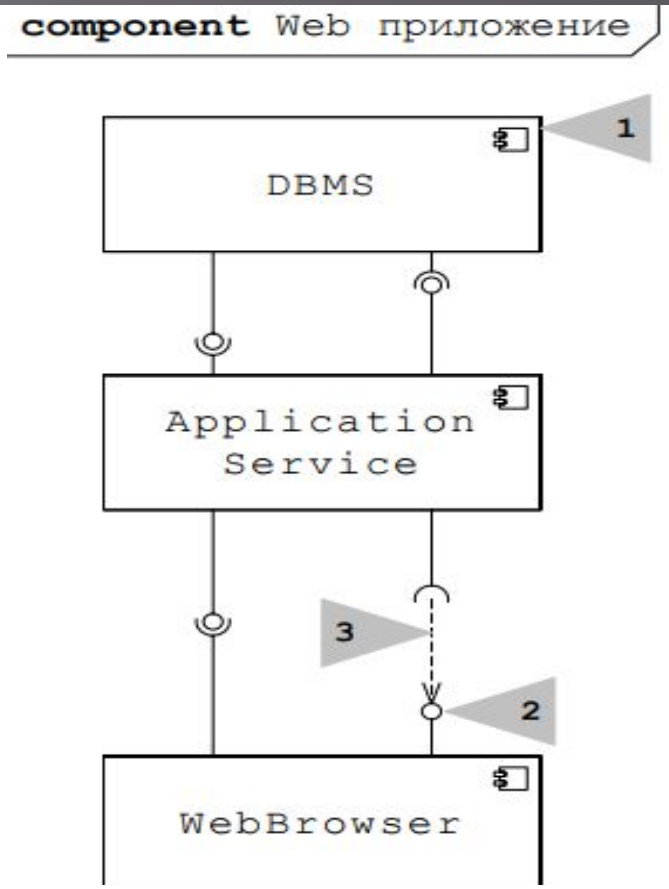
Діаграма компонентів (component diagram) - показує взаємозв'язок між модулями (логічними або фізичними), з яких складається моделююча система.



Нотація мови UML для компоненту



Діаграма компонентів для системи «Магазин»



Нотація діаграми компонентів

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Виходячи із вище дослідженого і розглянутого матеріалу створюємо реляційну базу даних — база даних, основана на реляційній моделі даних.

Основні компоненти реляційного відношення

СТВОРЕНА БАЗА ДАНИХ

На відміну від ієрархічної і сітьової моделей даних в реляційної відсутнє поняття групового відношення. Для відображення асоціацій між кортежами різних відносин використовується дублювання їх ключів. Атрибути, що представляють собою копії ключів інших відношень, називаються зовнішніми ключами. У своїй БД я використав тип зв'язку між сутностями один до багатьох та один до одного.

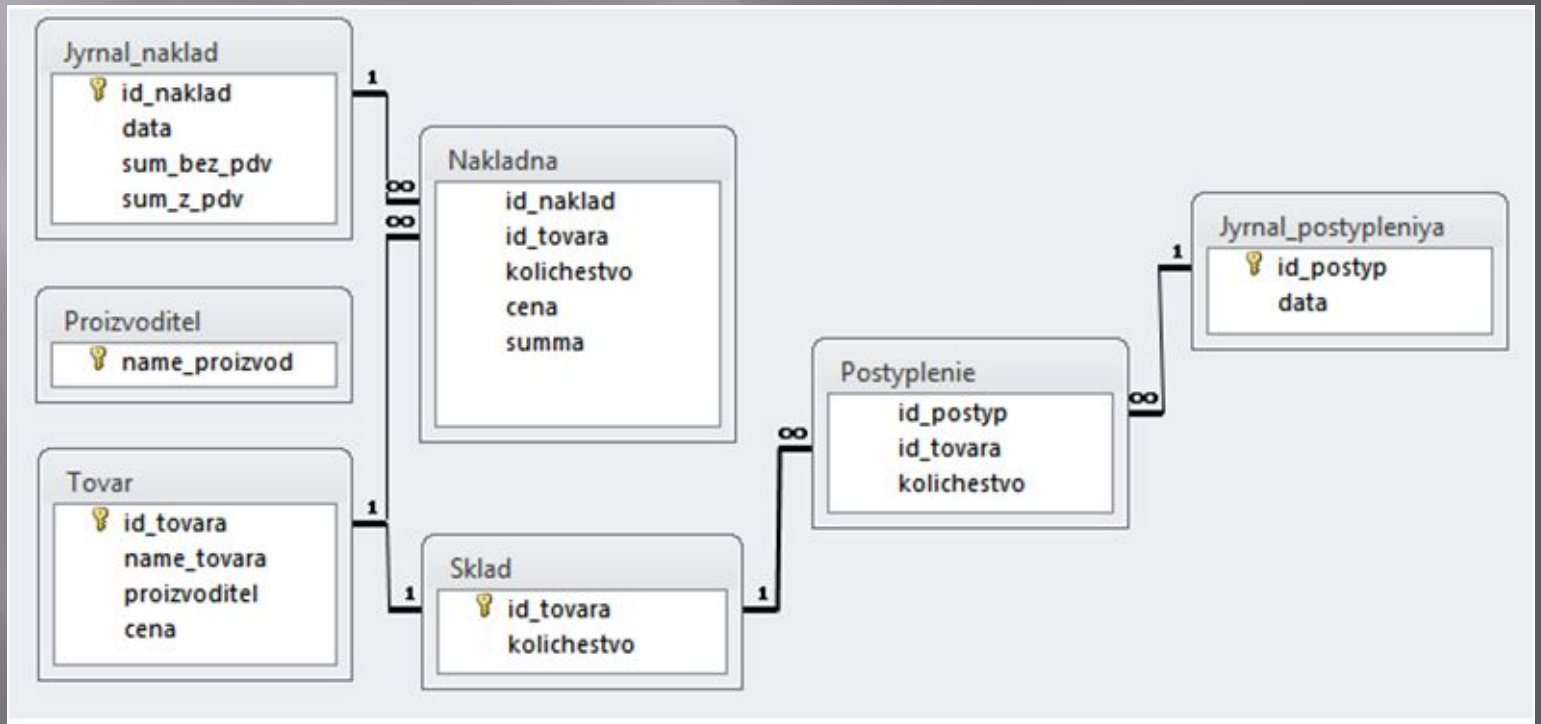
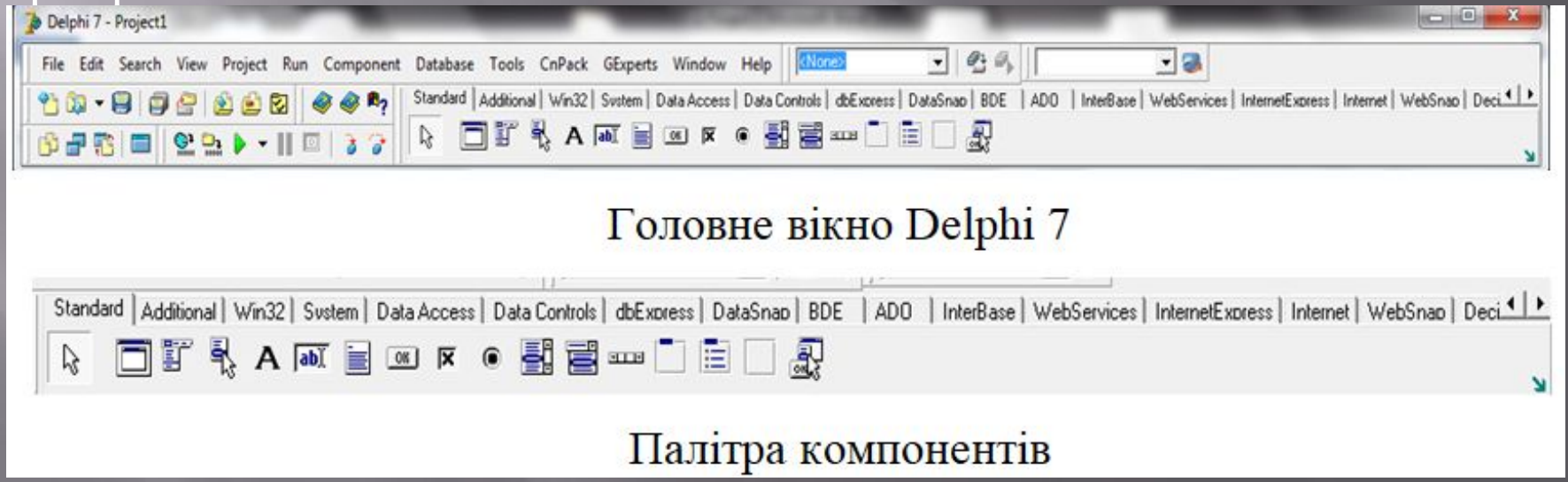


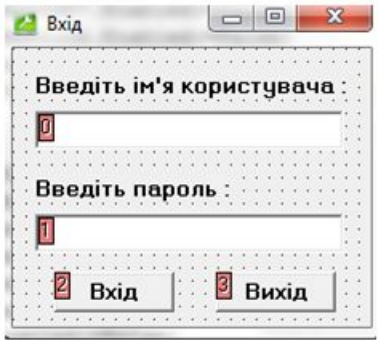
Схема зв'язків між таблицями в базі даних

ПРОГРАМНЕ СЕРЕДОВИЩЕ

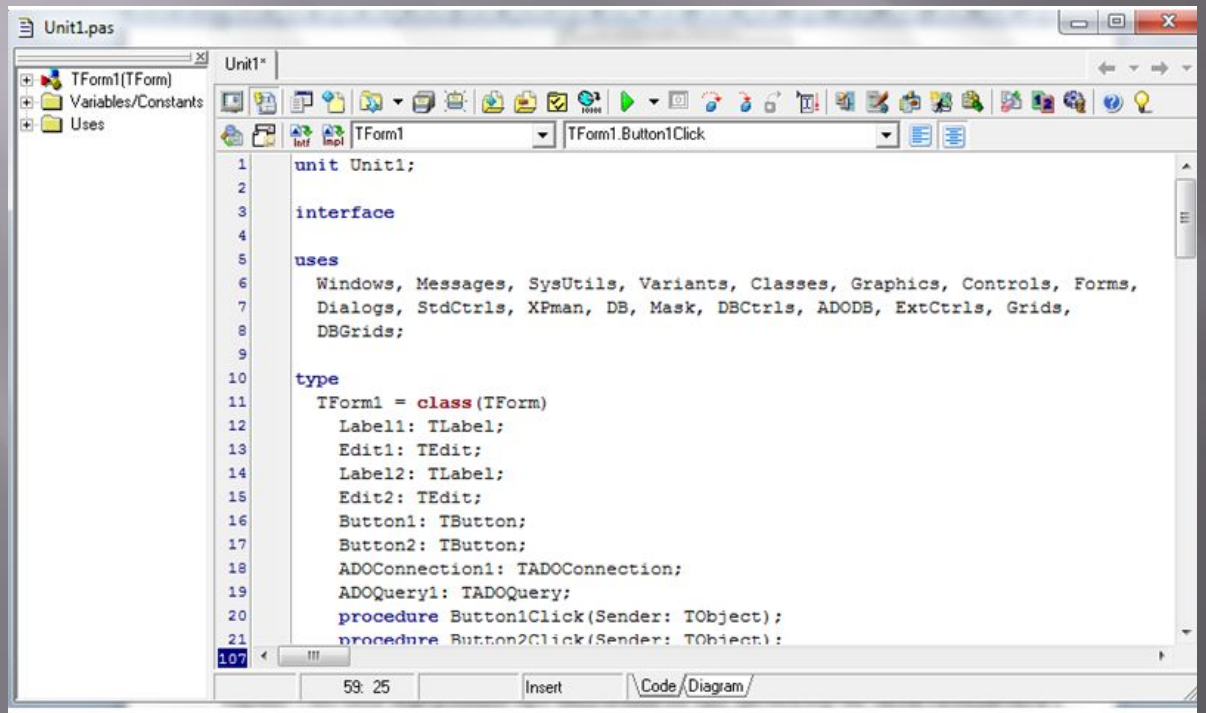
Delphi — мова програмування, що ґрунтується на діалекті мови Pascal. Для створення інтерфейсу і зв’язку з БД(бази даних) використаємо Delphi 7, тому що в Delphi 7 легко створювати дружній інтерфейс, є підтримка мови програмування objective pascal а також є багато додаткових компонентів для роботи з БД, що значно полегшують розробку програмного забезпечення.



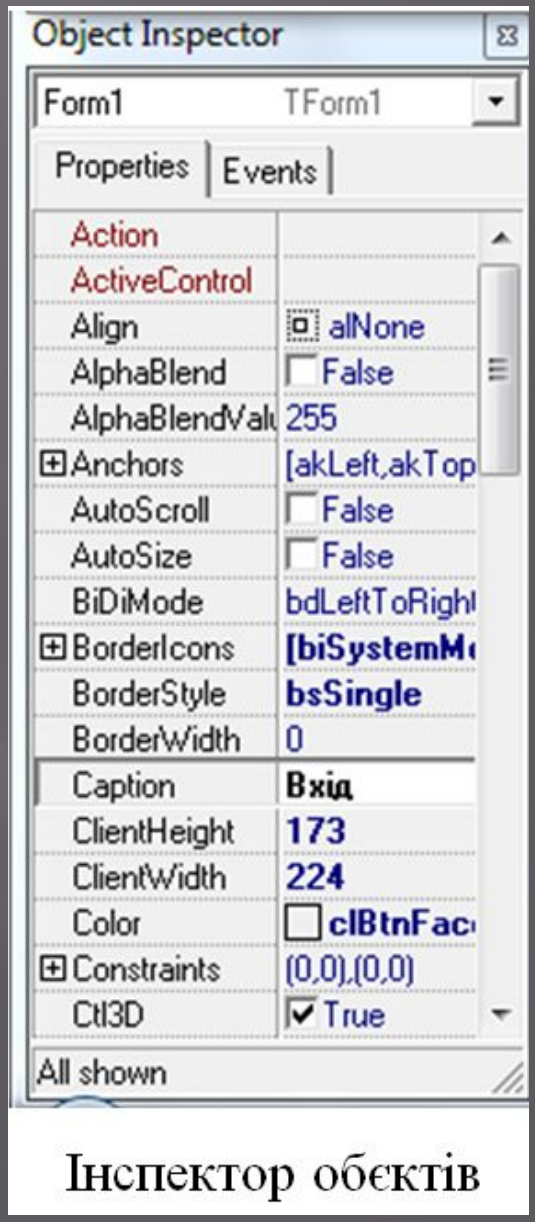
СТВОРЕННЯ ФОРМИ



Форма програми з компонентами



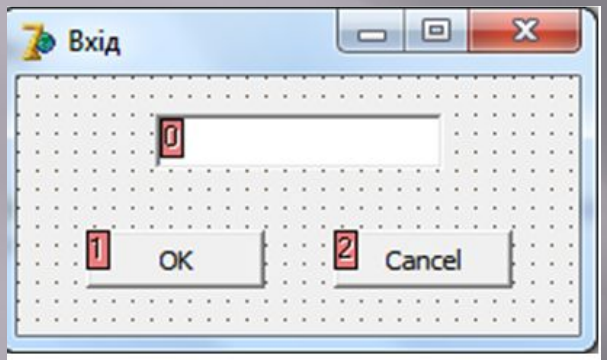
Вікно коду



Інспектор обєктів

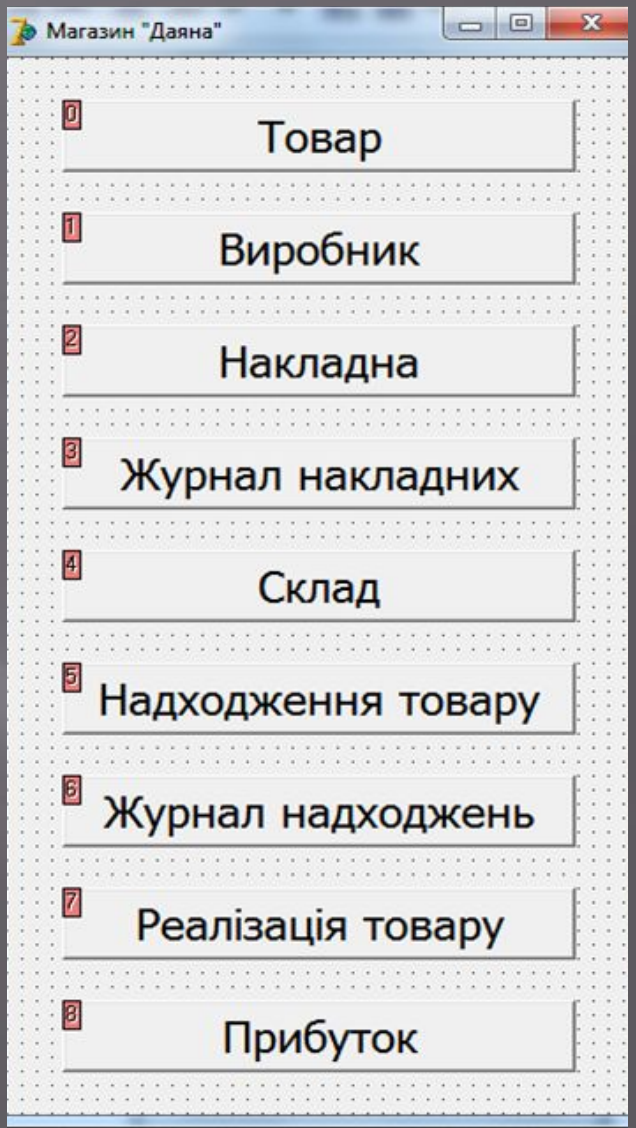
РОЗРОБКА ІНТЕРФЕЙСУ КОРИСТУВАЧА

За допомогою форми «UnitPass» відбувається вхід в програму



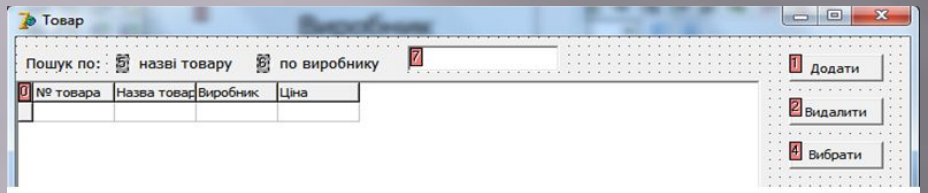
Форма «UnitPass»

Головна форма програми «UnitMain» дозволяє виконувати різні функції: перегляд товару, переглянути виробників, створити накладну, переглянути журнал накладних, переглянути кількість товару на складі, провести надходження товару, переглянути журнал надходжень, проаналізувати реалізацію товару а також подивитися прибуток

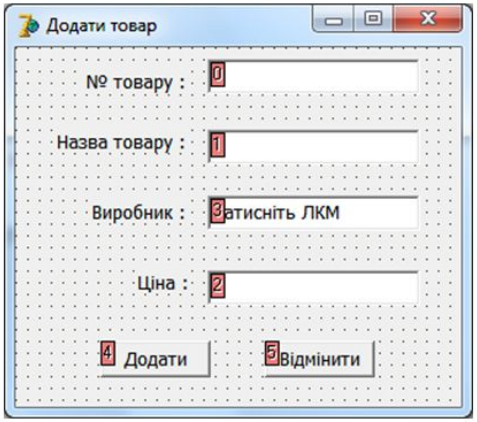


Форма «UnitMain»

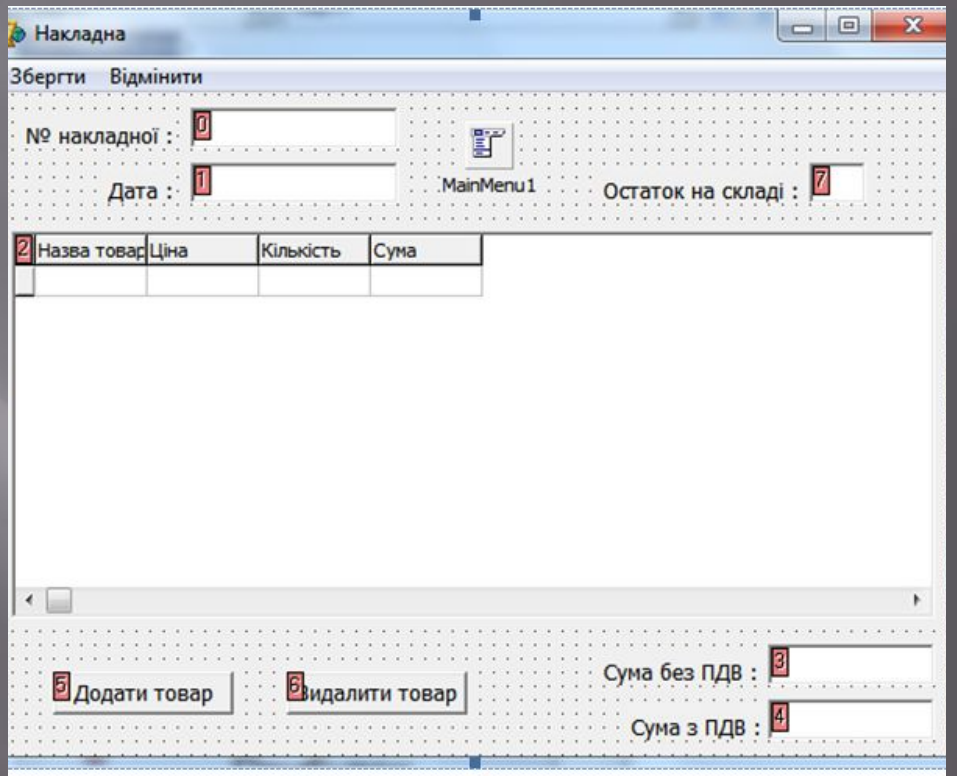
ПЕРЕГЛЯНЕМО ЯК ПРАЦЮЮТЬ КІЛЬКА СТВОРЕНИХ ФОРМ



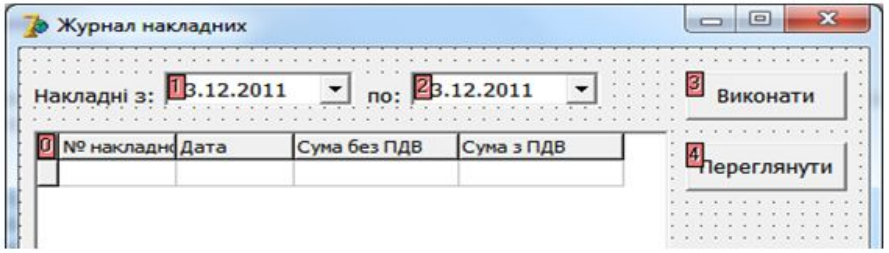
Форма «UnitTovar»



Форма «UnitAddTovar»



Форма «UnitNakladna»



Форма «UnitJyrnal»

ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Тестування - процес виконання програми з метою виявлення помилок. Після збору і оцінювання результатів тестування починається відображення якості та надійності ПЗ. Існують такі види тестування:

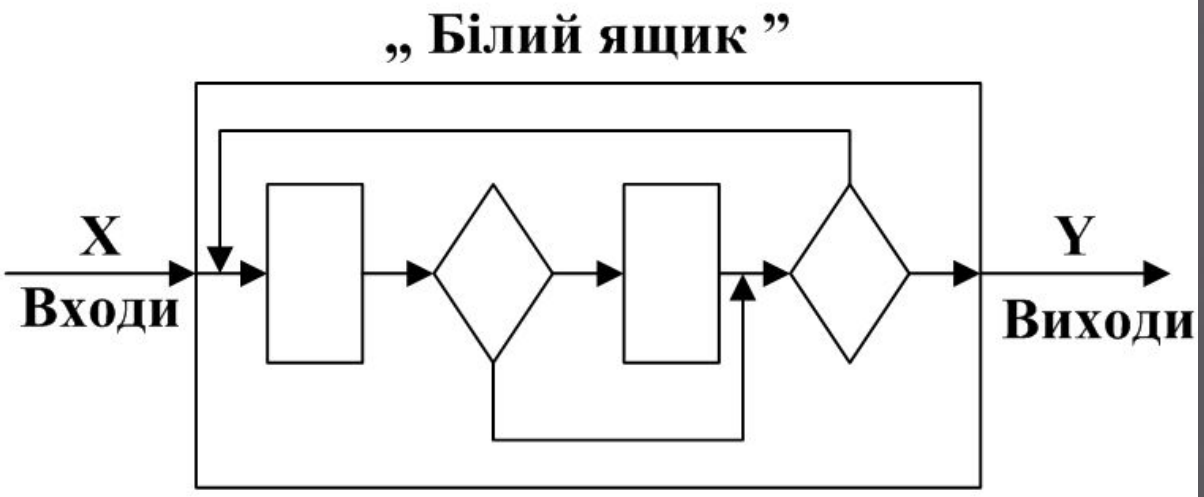
- *Модульне тестування*
- *Інтеграційне тестування*
- *Системне тестування*
- *Вихідне тестування*
- *Приймальне тестування*
  - структурне тестування (тестування «білого ящика»);

Інформаційні потоки процесу тестування

СТРУКТУРНЕ ТЕСТУВАННЯ “БІЛОГО ЯЩИКА”

Зазвичай тестування «білого ящика» засноване на аналізі керуючої структури програми. Програма вважається повністю перевіреної, якщо проведено вичерпне тестування маршрутів (шляхів) її графа управління. У цьому випадку формуються тестові варіанти, в яких:

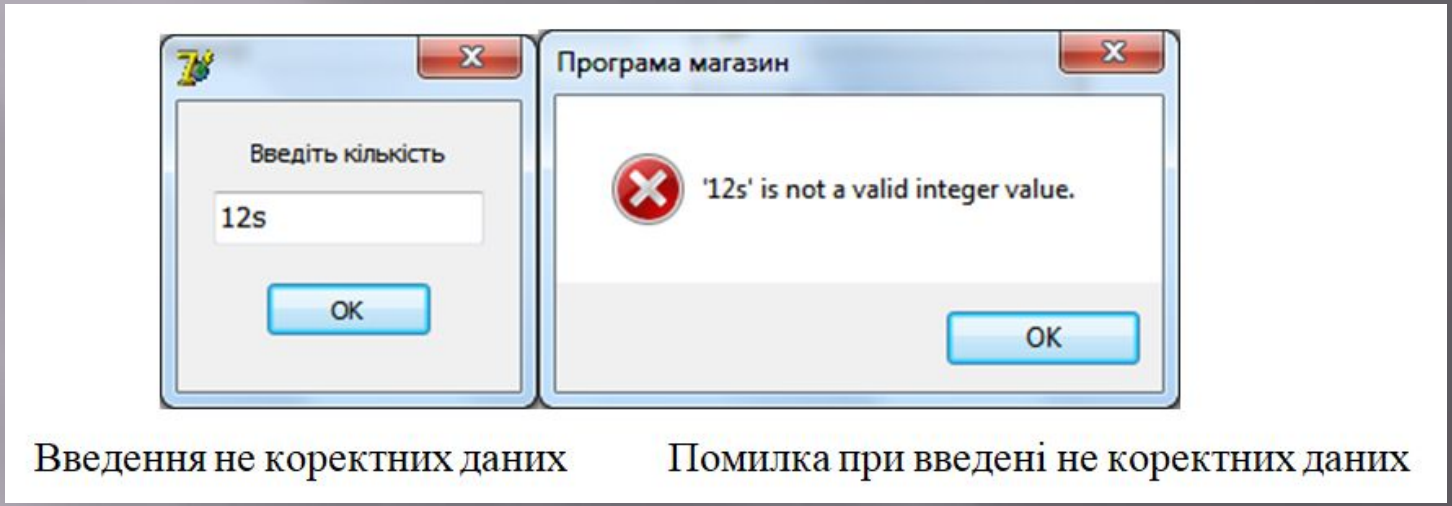
- *гарантується перевірка всіх незалежних маршрутів програми;*
- *проходяться гілки True, False для всіх логічних рішень;*
- *виконуються всі цикли (у межах їх меж і діапазонів);*
- *аналізується правильність внутрішніх структур даних.*



Тестування «білого ящика»

ТЕСТУВАННЯ ПРОГРАМНОГО КОДУ

Тестування на введення даних. При введені кількості товару можна випадково ввести букву і після цього в програмі виникає помилка .



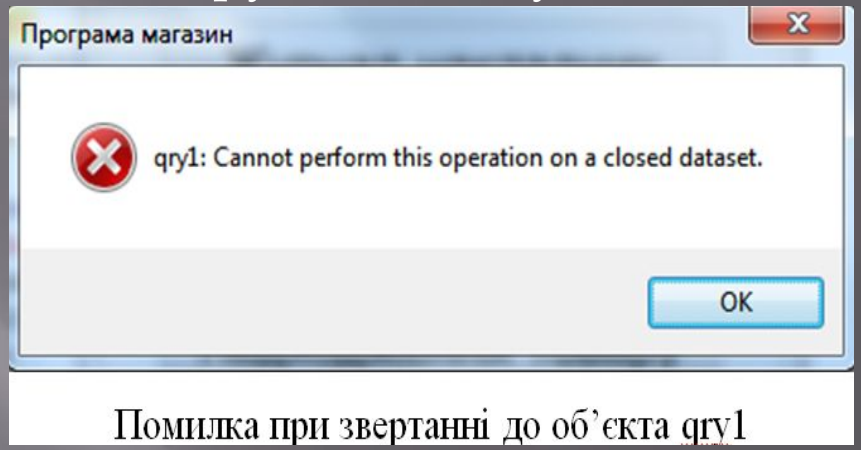
Введення не коректних даних Помилка при введені не коректних даних

Ця проблема вирішується за допомогою процедури на натиснення клавіші, тобто в поле можна ввести лише цифри , а «#8» це код натиснення клавіші «Backspace», для того щоб можна було редагувати кількість. Програмно це має наступний вигляд:

```
procedure TfrmKolichestvo.edtKolichestvoKeyPress(Sender: TObject;
  var Key: Char);
begin
if Not (Key in ['0'..'9', #8])then Key:=#0;
end;
end.
```

ТЕСТУВАННЯ «БІЛОГО ЯЩИКУ»

Помилка виникає в процедурі створення накладної. Помилка виникає тому що в процедурі «TfrmNakladna.FormShow» при звертанні до об'єкта qry1 не має даних, це викликано там що об'єкт qry1 не виконує запит що був йому переданий.



Помилка при звертанні до об'єкта qry1

Програмно це виглядає так:

```
procedure TfrmNakladna.FormShow(Sender: TObject);
var date: TDateTime;
begin
DataModule1.qry1.SQL.Clear;
DataModule1.qry1.SQL.Text:='SELECT * FROM jyrnal_naklad';
nomnaklad.Text:= IntToStr( DataModule1.qry1.RecordCount +1);
date:=Now;
data.Text:=FormatDateTime('dd.mm.yy', date);
nomnaklad.ReadOnly:=True;
data.ReadOnly:=True;
DataModule1.qry1.SQL.Clear;
DataModule1.qryNakladna.SQL.Clear;
DataModule1.qryNakladna.SQL.Text:='SELECT  tovar.id_tovara, tovar.name_tovara,  nakladna.cena, nakladna.kolichestvo,
nakladna.summa'+
                    ' FROM nakladna, tovar WHERE nakladna.id_tovara=tovar.id_tovara AND nakladna.id_naklad='+nomnaklad.Text;
DataModule1.qryNakladna.Active:= True;
end;
```

Тому потрібно знайти рядок запиту qry1 і після нього активувати запит:

```
procedure TfrmNakladna.FormShow(Sender: TObject);
var date: TDateTime;
begin
DataModule1.qry1.SQL.Clear;
DataModule1.qry1.SQL.Text:='SELECT * FROM jyrnal_naklad';
DataModule1.qry1.Active:= True;
…
end;
```

ВИСНОВКИ:

Згідно з сформованих задач розглянуто та досліджено спрогнозовану роботу створенної програми та системних моделей, що отримані та завдяки наступним крокам:

- в статті описан процес отримання системної моделі, форми програми та бази даних;

- отримано форму, що задається зовнішніми критеріями, і яка вже досліджена на наявність помилок прогнозуючого процесу;

- для спрощення обробки моделей будь-якої складності розроблено технологію обробки інформації завдяки програмному алгоритму із зручним інтерфейсом користувача, що дозволяє не тільки автоматизувати процес обчислювання моделей, але й зберігати всю необхідну інформацію по отриманню навчених моделей різної складності.

ДЯКУЮ ЗА УВАГУ