МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

математичного факультету

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

**Методичні вказівки**

до Виконання лабораторних робіт з дисципліни

 Проектування програмних систем »

для студентів спеціальності

«Комп ' ютерні науки»

**Лабораторна я ра бота № 1**

**Дослідження предметної області**

**Мета :**навчитися визначати призначення , цілі , завдання , акторів і їх прецеденти для інформаційної системи , що проектується .

**Теоретичні відомості**:

Поняття предметної області бази даних є одним з базових понять інформатики і не має точного визначення. Його використання в контексті ІС передбачає існування стійкої в часі співвіднесеності між іменами, поняттями і певними реаліями зовнішнього світу, що не залежить від самої ІС і її кола користувачів. Таким чином, введення в розгляд поняття предметної області бази даних обмежує і робить доступним для огляду простір інформаційного пошуку в ІС і дозволяє виконувати запити за кінцевий час.

Ми під описом предметної області будемо розуміти опис оточення розроблюваної системи, типи користувачів системи, при цьому також вкажемо основні завдання, вирішення яких покладаються на систему.

У попередньому описі предметної області вводяться основні терміни (словник системи), визначаються типи користувачів і їх права, формулюються задачі, які повинна вирішувати розробляється система. При цьому при описі передбачається використовувати кошти звичайного мови і стандартної ділової графіки (Рисюнки, діаграми, таблиці).

При розробці словника системи необхідно визначити імена сутностей ( "студент", "викладач", "дисципліна"). При цьому термін сутність розуміється нами як компонент моделі предметної області, тобто як уже виділений на концептуальному рівні об'єкт. Кошти, виділені в предметної області об'єкти перетворюються аналітиком по суті.

Сутність є результатом абстрагування реального об'єкта. З об'єктами пов'язано дві проблеми: ідентифікація і адекватний опис. Для ідентифікації використовують ім'я, яке повинно бути унікальним. При цьому передбачається, що відбувається відмова від його змісту, який притаманний природної мови. Використовується тільки вказівний функція імені. Ім'я - це прямий спосіб ідентифікації об'єкта. До непрямих методів ідентифікації об'єкта відносять визначення об'єкта через його властивості (характеристики або ознаки).

Об'єкти взаємодіють між собою через свої властивості, що породжує ситуації. Ситуації - це взаємозв'язки, що виражають взаємини між об'єктами. Ситуації в предметної області описуються за допомогою висловлювань про предметну область. На даному етапі можна використовувати методи обчислення висловлювань і числення предикатів, тобто форРисьної, математичної логіки. Наприклад, висловлювання "Програміст і менеджер є службовці компанії" описує відношення включення. Таким чином, вся інформація про об'єкти і сутності предметної області описується за допомогою тверджень на природній мові.

Можна вказати структурні зв'язки, виділити статичні і динамічні ситуації (таким чином ввести в модель параметр часу), проте для детального опрацювання моделі зручніше використовувати розвинені засоби опису предметної області, наприклад засоби мови UML .

Отже, ставиться завдання розробити систему "АРМ секретаря кафедри" яка дозволяла б вести автоматизований облік даних про співробітників і студентів кафедри, забезпечувала гнучкі можливості при вирішенні запланованих і незапланованих специфічних завдань обробки облікових даних.

В рамках вирішення завдання розробки АРМ секретаря кафедри виділимо наступні сутності:

**викладачі -**викладачі кафедри;

**студенти**- учні вузу даної спеціальності;

студенти вчаться в **групах**, **група**є організуючою (об'єднавчої) сутністю для студентів;

**аспіранти**, мають ту особливість, що з одного боку самі можуть вести заняття, з іншого, самі є учнями і мають наукового керівника;

**дисципліна**- преподаваемая дисципліна (предмет, курс).

Наведені суті мають ряд атрибутів, які ми визначимо в подальшому.

Ведемо два типи користувачів: рядовий **користувач**(надалі **користувач**, і **адміністратор**. Передбачається, що **користувач**може звертатися до системи із запитом, виводити звіти, **адміністратор**додатково може модифікувати дані. Наприклад, в ролі користувача може виступати помічник секретаря кафедри, в ролі адміністратора сам секретар, або відповідальний викладач.

З урахуванням введених термінів розробляється систем повинна забезпечувати:

організацію повного та достовірного обліку всіх співробітників і студентів кафедри;

інформаційну підтримку прийнятих управлінських рішень, формування повної і достовірної інформації про освітні процеси і результати діяльності кафедри;

скорочення трудовитрат на підготовку первинних документів і звітів;

усунення дублювання при введенні інформації і, що виникають при цьому механічних помилок;

зручний інтерфейс користувача;

розмежування повноважень рядових користувачів і адміністратора.

В даному прикладі ми вирішуємо приватну задачу - розробляємо АРМ секретаря кафедри, тому структурною одиницею найвищого рівня для нас приймається кафедра, яку ми будемо мати на увазі за замовчуванням, тобто передбачається, що всі елементи моделі відноситься тільки до даної кафедрі, що явно не специфицируется . Структури більш високого рівня, такі як факультет, вуз, нами розглядатися не будуть.

**Предметна область:**

Компанія є мережею магазинів роздрібної торгівлі або салонів, що надають різні послуги (див. Варіант) . У компанії є свій сайт, де представлена інформація по продаваним товарам або послугам . Але сайт є лише інформаційним і не дозволяє проводити покупку товарів онлайн або надавати послуги . У компанії відсутня можливість оперативного відстеження зміни кількості товарів в наявності в магазинах або на складі .

Компанії потрібно ІС роздрібної торгівлі або надання послуг , яка повинна автоматизувати наступні процеси:

         1. Обслуговування клієнтів, включаючи оформлення та оплату замовлень або послуг для юридичних і фізичних осіб про н лайн.

        2. Управління мережею магазинів або салонів .

        3.Управління запасами товарів або матеріалів для надання послуг.

**завдання :**

Дослідити предметну область інформаційної системи , сформулювати призначення , цілі та завдання , які необхідно виконати для її реалізації .

Визначити діючих акторів та прецеденти їх взаємодії з системою .

Заповнити таблицю :

|  |  |
| --- | --- |
| актори | прецеденти |
|  |  |
|  |  |

**Варіанти :**

Вари ант 1: ІС по продажу комп'ютерів .

Варіант 2. ІС з прокату автомобілів .

.

Варіант 3. ІС по прокату СД дисків .

Варіант 4. ІС по продажу медикаментів .

Варіант 5. ІС розрахунку обсягу матеріалів для виробництва меблів .

Варіант 6. ІС обліку замовлень в ательє .

Варіант 7. ІС обліку замовлень матеріалів для стоматологічного кабінету .

Варіант 8. ІС обліку закупівель і витрат сировини для проведення ремонтних робіт в квартирах.

Варіант 9. ІС прокат велосипедів .

Варіант 10. ІС по наданню послуг в салоні краси .

**Лабораторна робота №2**

**Розробка UML діаграм прецедентів**

**Мета :**ознайомитися з про засобами представлення основних завдань, що розробляється і дій, які вона повинна виконувати, навчитися разрабат и вать UML -діаграмми прецедентів.

**Теоретичні відомості:**

Мова UML є мовою специфицирования і візуалізації, основними одиницями його є діаграми.

Діаграма в UML - це графічне представлення набору елементів, зображуване найчастіше у вигляді пов'язаного графа з вершинами (сутностями) і ребрами (відносинами). Діаграми характеризують систему з різних точок зору. Діаграма - в деякому сенсі одна з проекцій системи. Як правило, діаграми дають згорнуте представлення елементів, з яких складена система. Один і той же елемент може бути присутнім у всіх діаграмах, або тільки в декількох (найпоширеніший варіант), або не бути присутнім в жодній (дуже рідко). Теоретично діаграми можуть містити будь-які комбінації сутностей і відносин. На практиці, однак, застосовується порівняно невелика кількість типових комбінацій, відповідних п'яти найбільш вживаним видів, які складають архітектуру програмної системи (див. Наступний розділ). Таким чином, в UML виділяють дев'ять типів діаграм:

Діаграма класів

діаграми об'єктів;

діаграми прецедентів;

діаграми послідовностей;

діаграми кооперації;

діаграми станів;

діаграми дій (діяльності);

діаграми компонентів;

діаграми розгортання.

Концептуальна модель UML

На діаграмі класів показують класи, інтерфейси, об'єкти і кооперації, а також їх відносини. При моделюванні об'єктно-орієнтованих систем цей тип діаграм використовують найчастіше. Діаграма класів відповідають статичному увазі системи з точки зору проектування. Діаграма класів, які включають активні класи, відповідають статичному увазі системи з точки зору процесів.

На діаграмі об'єктів представлені об'єкти і відносини між ними. Вони є статичними "фотографіями" екземплярів сутностей, показаних на діаграмах класів. Діаграми об'єктів, як і діаграми класів, відносяться до статичного виду системи з точки зору проектування або процесів, але з розрахунком на справжню або макетну реалізацію.

На діаграмі прецедентів представлені прецеденти і актори (окремий випадок класів), а також відносини між ними. Діаграми прецедентів відносяться до статичного виду системи з точки зору прецедентів використання. Вони особливо важливі при організації і моделюванні поведінки системи.

Діаграми послідовностей і кооперації є окремими випадками діаграм взаємодії. На діаграмах взаємодії представлені зв'язку між об'єктами; показані, зокрема, повідомлення, якими об'єкти можуть обмінюватися. Діаграми взаємодії відносяться до динамічного виду системи. При цьому діаграми послідовності відображають тимчасову упорядкованість повідомлень, а діаграми кооперації - структурну організацію обмінюються повідомленнями об'єктів. Ці діаграми є ізоморфними, тобто можуть бути перетворені одна в одну.

На діаграмах станів ( Statechart diagrams ) представлений автомат, що включає в себе стану, переходи, події і види дій. Діаграми станів відносяться до динамічного виду системи; особливо вони важливі при моделюванні поведінки інтерфейсу, класу або кооперації. Вони акцентують увагу на поведінці об'єкта, що залежить від послідовності подій, що дуже корисно для моделювання реактивних систем.

Діаграма діяльності - це окремий випадок діаграми станів; на ній представлені переходи потоку управління від однієї діяльності до іншої всередині системи. Діаграми діяльності відносяться до динамічного виду системи; вони найбільш важливі при моделюванні її функціонування і відображають потік управління між об'єктами.

На діаграмі компонентів представлена ​​організація сукупності компонентів і існуючі між ними залежності. Діаграми компонентів відносяться до статичного виду системи з точки зору реалізації. Вони можуть бути співвіднесені з діаграмами класів, так як компонент зазвичай відображається на один або кілька класів, інтерфейсів або кооперацій.

На діаграмі розгортання представлена ​​конфігурація обробних вузлів системи і розміщених в них компонентів. Діаграми розгортання відносяться до статичного виду архітектури системи з точки зору розгортання. Вони пов'язані з діаграмами компонентів, оскільки в вузлі зазвичай розміщуються один або кілька компонентів.

Тут наведено неповний список діаграм, що застосовуються в UML . Інструментальні засоби дозволяють генерувати і інші діаграми, наприклад, такі як діаграми профілю баз даних, діаграми WEB -додатків і т.д.

Моделювання починається з визначення основних завдань системи, що розробляється і дій, які вона повинна виконувати. Для цих цілей використовуються діаграми прецедентів. Як вже говорилося, на діаграмах прецедентів вказуються прецеденти і актори, а також відносини між ними.

***Прецедент***( Use case ) - це опис послідовності виконуваних системою дій, яка виробляє спостережуваний результат, значимий для якогось певного ***актора***( Actor ). Прецедент застосовується для структурування поведінкових сутностей моделі. Прецедент тільки декларує опис якої-небудь дії системи, відповідаючи на питання "що робити?", Але не вказує якими засобами. Конкретна реалізація складають специфікацію прецедентом поведінки забезпечується класом, кооперацією класів або компонентом.

Актор є чіткий безліч ролей, які користувачі прецедентів виконують під час взаємодії з ними. Зазвичай актор представляє роль, яку в даній системі грає людина, апаратний пристрій або навіть інша система. У системі "АРМ секретаря кафедри" акторами є адміністратор **(адмін)**і **користувач**.

Графічно прецедент зображується у вигляді обмеженого безперервної лінією еліпса, зазвичай містить лише його ім'я, актор має піктограму "чоловічок".

Для того, щоб побудувати діаграму прецедентів необхідно виявити елементарні дії, вироблені системою і зіставити їх з прецедентами. При цьому, імена прецедентів бажано давати так, щоб вони вказували на поведінку, часто такі імена містять дієслова, наприклад, "сформувати звіт", "знайти дані за критерієм" і т.д. Можна давати імена прецедентів іменниками, які передбачають деякі дії, наприклад, "авторизація", "пошук", "контроль".

Повертаючись до моделювання АРМ секретаря кафедри, виділимо прецеденти:

**Редагування даних**,

**Пошук студента**,

**Пошук викладача**,

**Видача списку дисциплін, що викладаються**,

**Авторизація**.

Елементи діаграми прецедентів (прецеденти і актори) повинні бути пов'язані відносинами.

Найбільш поширеним ставленням між прецедентами, прецедентами і акторами є асоціація. У деяких випадках можуть бути використані відносини узагальнення. Дані відносини мають таке ж значення, що і на діаграмі класів.

Крім того, між прецедентами в мові UML визначені дві специфічні залежності - відношення включення та відношення розширення.

Ставлення включення між прецедентами означає, що в деякій точці базового прецеденту інкорпороване (включено) поведінка іншого прецеденту. Включається прецедент ніколи не існує автономно, а інстанціруется тільки як частина осяжний прецеденту. Можна вважати, що базовий прецедент запозичує поведінку включаються. Завдяки наявності відносин включення вдається уникнути багаторазового опису одного і того ж потоку подій, оскільки загальна поведінка можна описати у вигляді самостійного прецеденту, що включається в базові. Ставлення включення є прикладом делегування, при якому ряд обов'язків системи описується в одному місці (у включається прецедент), а інші прецеденти, коли необхідно, включають ці обов'язки в свій набір.

Відносини включення зображуються у вигляді залежностей зі стереотипом " include ". Щоб уточняти місце в потоці подій, де базовий прецедент включає поведінку іншого, ви просто пишете слово include , за яким слідує ім'я включається прецеденту.

Ставлення розширення застосовують для моделювання таких частин прецеденту, які користувач сприймає як необов'язкове поведінку системи. Тим самим можна розділити обов'язкове і необов'язкове поведінку. Відносини розширення використовуються також для моделювання окремих субпотоків, виконуваних лише при певних обставинах. Нарешті, їх застосовують для моделювання декількох потоків, які можуть включатися в деякій точці сценарію в результаті явного взаємодії з актором.

Ставлення розширення зображують у вигляді залежності зі стереотипом " extend ". Точки розширення базового сценарію перераховуються в додатковому розділі. Вони є просто мітками, які можуть з'являтися в потоці базового прецеденту.

Прикладом використання даного відносини може бути доступ до бази даних має оперативну частину і архів. При цьому якщо запит забезпечується даними оперативної частини, виконується основний (базовий) доступ до даних, якщо ж даних оперативної частини недостатньо, проводиться доступ до даних архіву, тобто доступ йде по розширеному сценарієм.

У нашому випадку, прецедент **редагування даних**включає в себе прецеденти: **введення даних**, **видалення даних**, **зміна даних**.

Діаграма прецедентів АРМ секретаря кафедри показана на рис.1.



Рис. 1. Діаграма прецедентів АРМ секретаря кафедри

**Основні відомості про роботу в середовищі Rational**

**Rose**

Розглядається версія Rational Rose Enterprise Edition 2001®.

**1.1 Елементи екрану**

На рис. 2. показані різні частини інтерфейсу Rose.

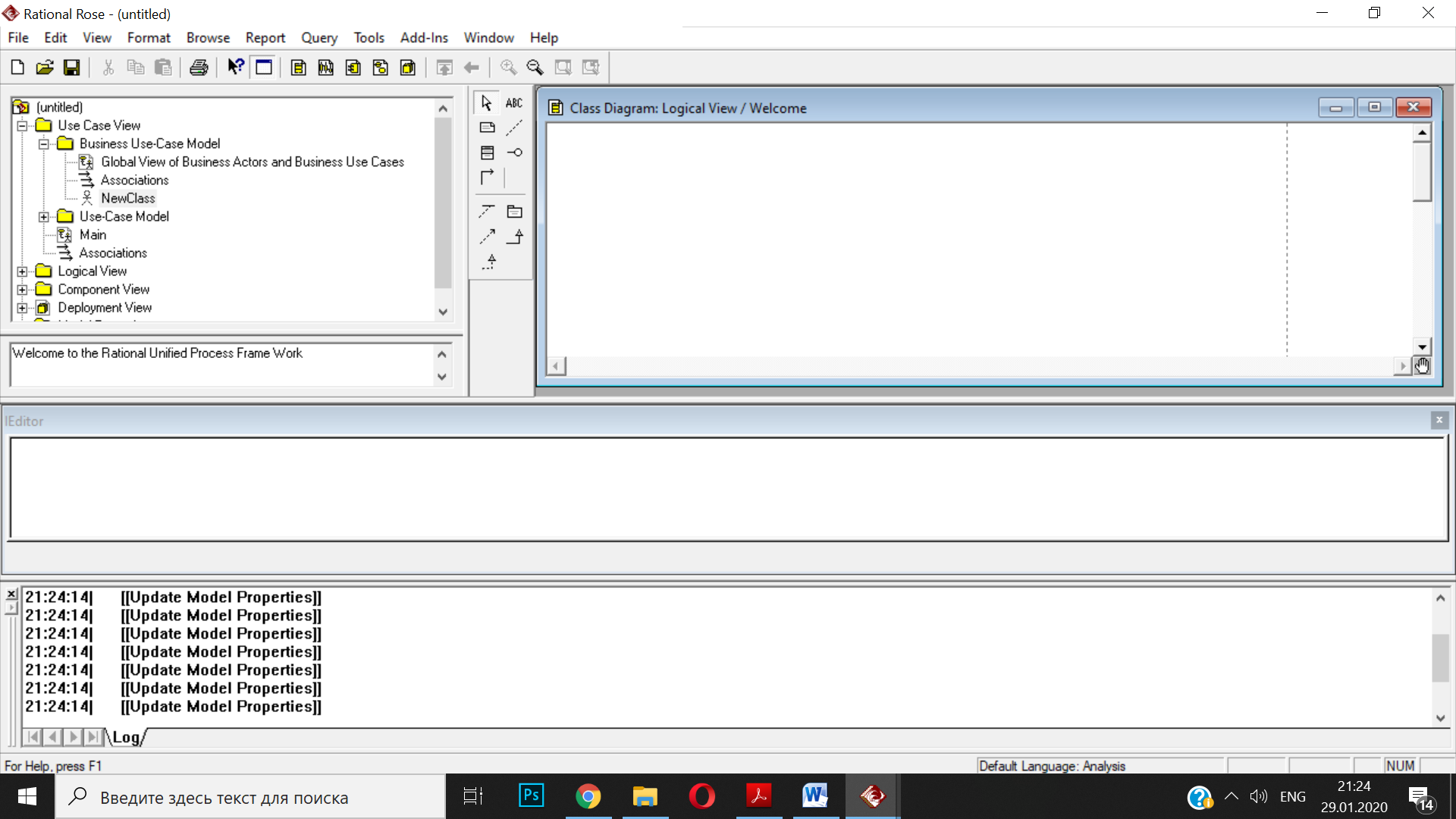


Рис. 2 . Інтерфейс Rose

Шість основних елементів інтерфейсу Rose - це браузер, вікно

документації, панелі інструментів, вікно діаграми і журнал. Їх призначення полягає в наступному.

✓ Браузер - використовується для швидкої навігації по моделі.

✓ Вікно документації - застосовується для роботи з текстовим

✓ описом елементів моделі.

✓ Панелі інструментів - застосовуються для швидкого доступу до

✓ поширеним командам.

✓ Вікно діаграм - використовується для перегляду і редагування

✓ діаграм UML.

✓ Журнал - застосовується для перегляду помилок і звітів про

✓ результатах виконання різних команд.

✓ Вікно редактора - редагує текстові дані.

*браузер*

Браузер - це ієрархічна структура, що дозволяє здійснювати

навігацію по моделі. Все, що додається до неї - дійові особи,

варіанти використання, класи, компоненти, пакети - буде показано у вікні

браузера.

За допомогою браузера можна:

• додавати до моделі елементи (дійові особи, варіанти

використання, класи, пакети, компоненти, діаграми, пакети і

т.д.),

• переглядати існуючі елементи моделі,

• переглядати існуючі зв'язки між елементами моделі,

• переміщати елементи моделі,

• перейменовувати елементи,

• додавати елементи моделі до діаграми,

• пов'язувати елемент з файлом або адресою Інтернет,

• групувати елементи в пакети,

• працювати з деталізованою специфікацією елемента,

• відкривати діаграму.

Браузер підтримує чотири вистави: уявлення варіантів

використання, логічне подання, подання розміщення і

уявлення компонентів.

Браузер організований в деревовидному стилі. Кожен елемент моделі

може містити інші елементи, що знаходяться нижче його в ієрархії. Знак "-"

близько елемента означає, що його гілка повністю розкрита. Знак "+" - що його

гілка згорнута.

*вікно документації*

З його допомогою можна документувати елементи моделі Rose.

Наприклад, можна зробити короткий опис кожної дійової особи. при

документуванні класу все, що буде написано в вікні документації,

з'явиться потім як коментар в створеному коді, що позбавляє від

необхідності згодом вносити ці коментарі вручну.

Документація буде виводитися також в звітах, що створюються в середовищі Rose.

У моделі Rose підтримується чотири вистави - уявлення

варіантів використання (Use Case View), логічне уявлення (Logical

View), уявлення компонентів (Component View) і уявлення

розміщення (Deployment View). Кожне з них призначений для своїх цілей.

*Подання варіантів використання*

Це уявлення містить модель бізнес-процесів і модель

варіантів використання. На рис. 1 показано, як виглядає уявлення

варіантів використання в браузері Rose.

*логічне уявлення*

Логічне уявлення концентрується на тому, як система буде

реалізовувати поведінку, описане в варіантах використання. Воно дає докладну картину складових частин системи і описує взаємодію цих частин. Логічне уявлення включає в основному класи і діаграми класів. З їх допомогою конструюється детальний проект створюваної системи.

Логічне уявлення містить компоненти.

Класи.

Діаграма класів. Як правило, для опису системи використовується кілька діаграм класів, кожна з яких відображає певний підмножина всіх класів системи.

Діаграми взаємодії, що застосовуються для відображення об'єктів, що беруть участь в одному потоці подій варіанту використання.

Діаграми станів.

Пакети, які є групами взаємопов'язаних класів.

*подання компонентів*

Подання компонентів містить:

Компоненти, які є фізичними модулями коду.

Діаграми компонентів.

Пакети, які є групами пов'язаних компонентів.

*подання розміщення*

Остання вистава Rose - це уявлення розміщення. воно

відповідає фізичного розміщення системи, яке може відрізнятися від

її логічної архітектури.

У уявлення розміщення входять:

Процеси, які є потоками (threads), виконуваними в

відведеної для них області пам'яті.

Процесори, що включають будь-які комп'ютери, здатні

опрацьовувати дані. Будь-який процес виконується на одному або

декількох процесорах.

Пристрої, тобто будь-яка апаратура, не здатна обробляти

дані. До числа таких пристроїв відносяться, наприклад, термінали

введення-виведення і принтери.

Діаграма розміщення.

**Створення моделі варіантів використання**

При запуску Rational Rose в вікні **Create New Model**виберіть варіант

Rational Unified Process (Раціональний Уніфікований процес)

і натисніть ОК (рис. 3). В результаті екран Rose набуде вигляду, показаний на рис. 2.

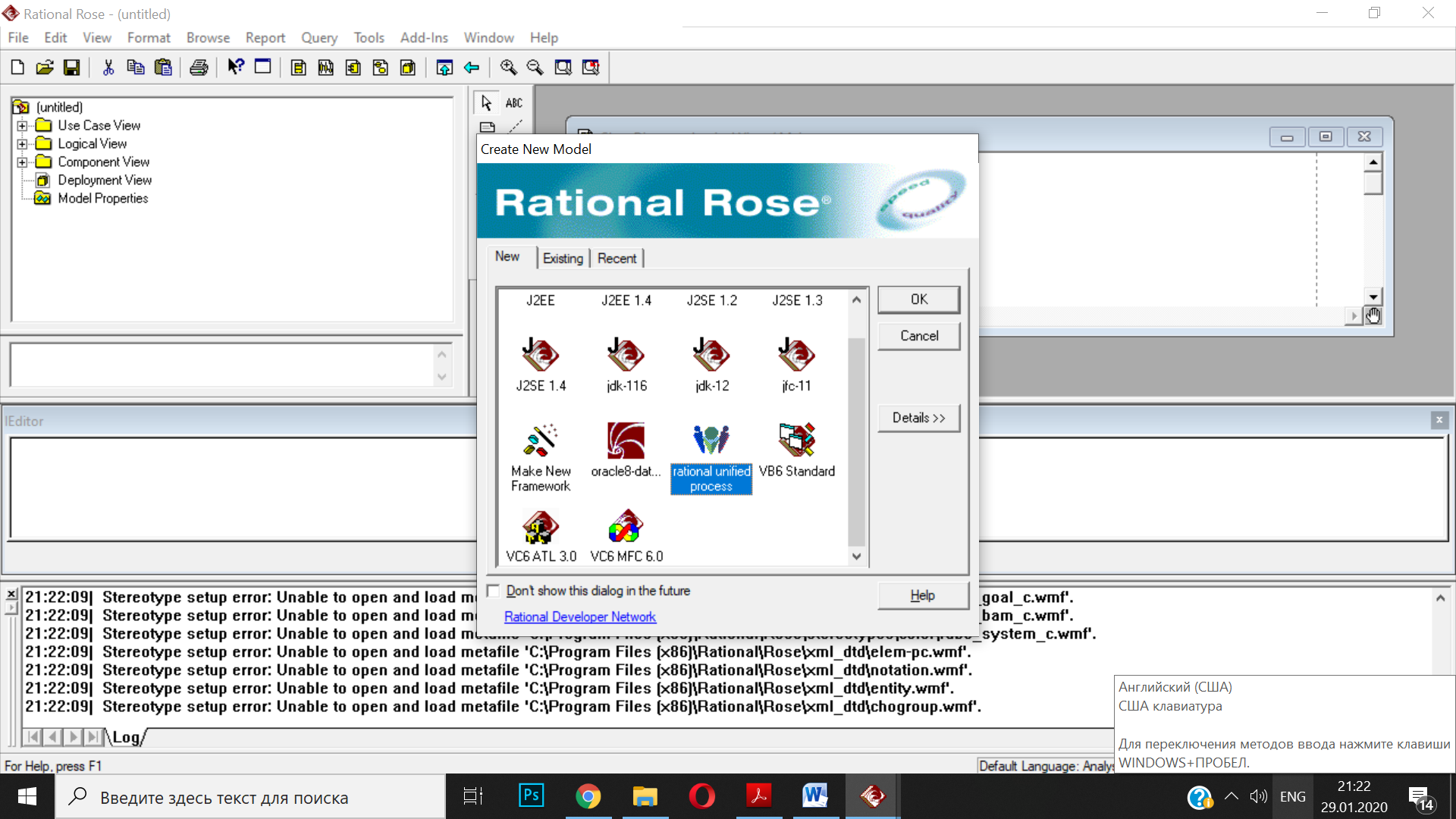


Рис. 3. Вікно вибору шаблону моделі

**Створення дійових осіб**

Щоб помістити дійова особа в браузер, виконайте наступне.

1. Клацніть правою кнопкою миші на пакеті

Business Use Case Model уявлення UseCase View в

браузері.

2. Виберіть в меню пункт *New / Actor*.

3. У браузері з'явиться нова дійова особа під назвою

NewClass. Зліва від його імені ви побачите піктограму

дійової особи в вигляді чоловічка.

4. Виділивши нова дійова особа, введіть його ім'я

**Створення варіантів використання**

Щоб помістити варіант використання в браузер, виконайте

наступне.

1. Клацніть правою кнопкою миші на пакеті

Business Use Case Model уявлення Use Case View в

браузері.

2. Виберіть в меню пункт *New / Use Case*.

3. Новий варіант використання під назвою NewUseCase з'явиться

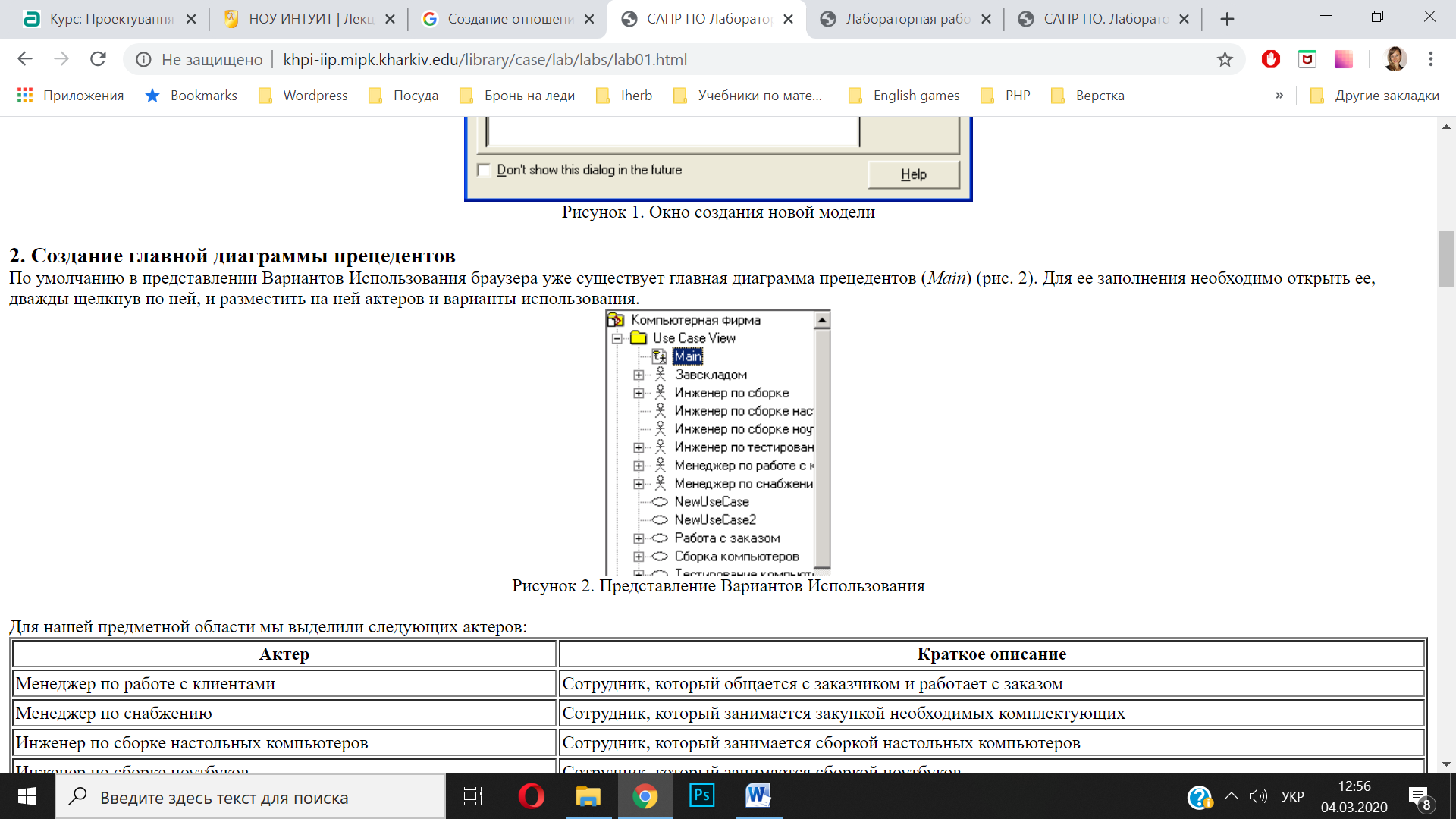
в браузері. Зліва від нього буде видно піктограма варіанти

використання UML у вигляді еліпса.

4. Виділивши новий варіант використання, введіть його назву

**Приклад виконання роботи.**

**1 . Створення головної діаграми прецедентів**  
За замовчуванням в поданні Варіантів Використання браузера вже існує головна діаграма прецедентів ( *Main*) (рис. 2). Для її заповнення необхідно відкрити її, двічі клацнувши по ній, і розмістити на ній акторів і варіанти використання.

  
Рис 2. Подання Варіантів Використання

Для нашої предметної області ми виділили наступних акторів:

|  |  |
| --- | --- |
| **актор** | **Короткий опис** |
| Менеджер по роботі з клієнтами | Співробітник, який спілкується з замовником і працює з замовленням |
| Менеджер з постачання | Співробітник, який займається закупівлею необхідних комплектуючих |
| Інженер по збірці настільних комп'ютерів | Співробітник, який займається складанням настільних комп'ютерів |
| Інженер по збірці ноутбуків | Співробітник, який займається складанням ноутбуків |
| Інженер з тестування | Співробітник, який займається тестуванням зібраного власноруч |
| завскладом | Співробітник, який завідує складом комплектуючих |

Розглянемо тепер, які можливості має надавати наша система:

* актор *Менеджер по роботі з клієнтами*використовує систему для оформлення, редагування замовлень і управління інформацією про клієнтів підприємства;
* актор *Менеджер з постачання*використовує систему для перегляду переліку необхідних для закупівлі комплектуючих і ведення інформації про постачання;
* актор *Інженер по збірці настільних комп'ютерів*використовує систему для перегляду нарядів на збірку персональних комп'ютерів, для замовлення комплектуючих зі складу та позначки про хід виконання роботи;
* актор *Інженер по збірці ноутбуків*використовує систему для перегляду нарядів на збірку ноутбуків, для замовлення комплектуючих зі складу та позначки про хід виконання роботи;
* актор *Інженер з тестування*використовує систему для перегляду нарядів на тестування зібраної продукції і позначки про хід виконання роботи;
* актор *Завскладом*використовує систему для обліку надходження і видачі комплектуючих.

На підставі вищевикладеного можна виділити наступні прецеденти:

|  |  |
| --- | --- |
| **прецедент** | **Короткий опис** |
| Робота із замовленням | Запускається менеджером.по роботі з клієнтами. Дозволяє вносити, змінювати, видаляти або переглядати замовлення. |
| Управління інформацією про клієнта | Запускається менеджером по роботі з клієнтами. Дозволяє додавати, змінювати або видаляти клієнтів, а також переглядати інформацію про клієнтів. |
| Управління інформацією про постачальників | Запускається менеджером з постачання. Дозволяє додавати, змінювати або видаляти постачальників, а також переглядати інформацію про постачальників. |
| Управління інформацією про комплектуючих | Запускається менеджером з постачання. Дозволяє переглядати інформацію про комплектуючих, проводити аналіз їх витрачання, прогнозувати необхідну їх кількість і робити замовлення. |
| збірка комп'ютерів | Запускається інженером по збірці. Дозволяє переглядати наряди на складання комп'ютерів і робити позначки про хід виконання роботи. |
| Вимога необхідних комплектуючих | Запускається інженером по збірці. Призначено для затребування необхідних комплектуючі зі складу. |
| тестування комп'ютерів | Запускається інженером з тестування. Дозволяє переглянути список комп'ютерів, що підлягають тестуванню та зробити позначки про хід виконання робіт. |
| Облік надходження та видачі комплектуючих | Запускається завскладом. Дозволяє вести облік надходження та видачі запчастин і комплектуючих. |

Створена головна діаграма прецедентів показана на рис. 3:

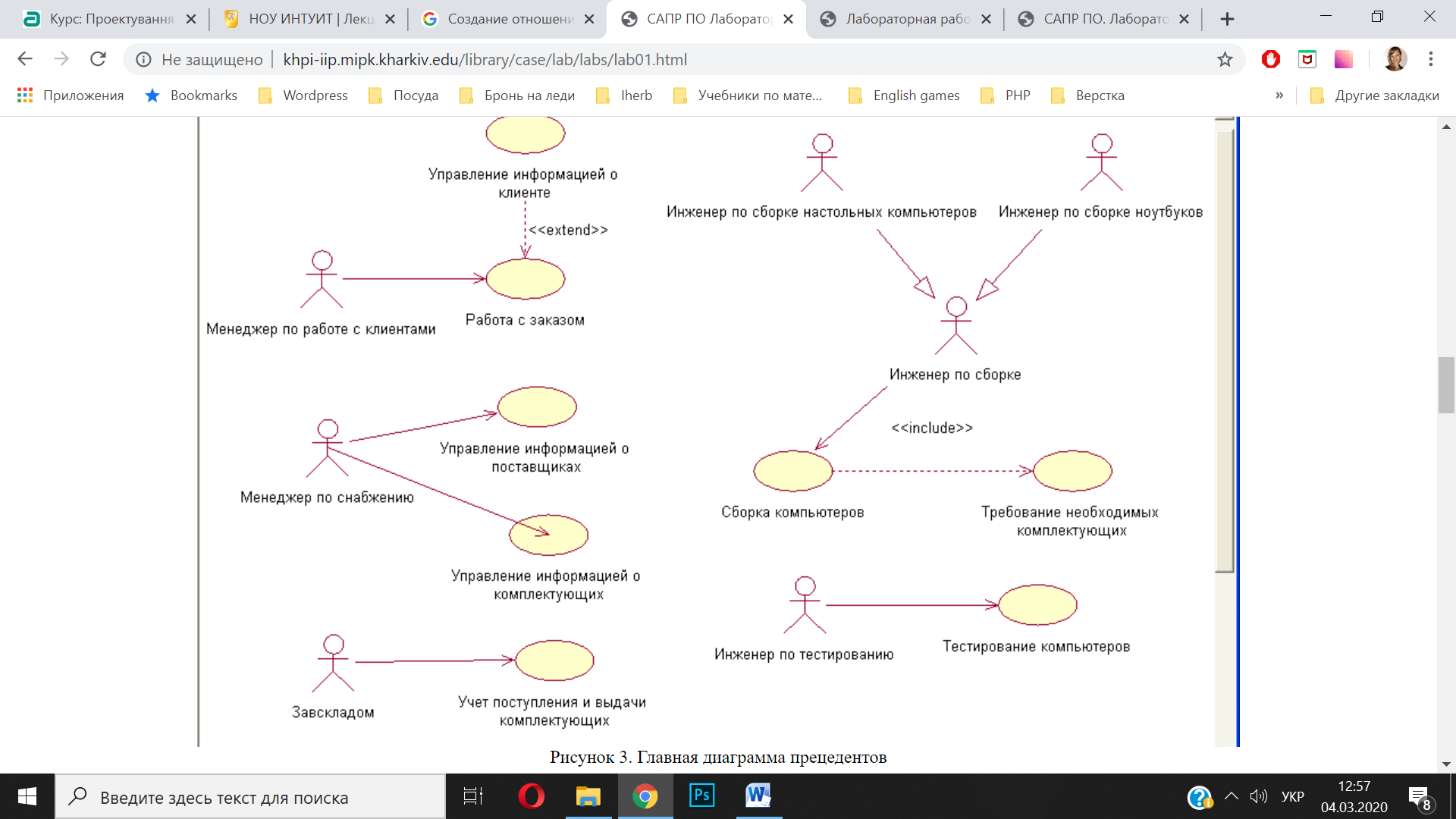


Рис 3. Головна діаграма прецедентів

Розглянемо тепер відносини між акторами і прецедентами. У мові UML можливий тільки один тип відносин між актором і прецедентом - відношення комунікації. Тому всіх актором ми зв'язали з прецедентами ставленням *Unidirectional Association*. Оскільки інший тип відносин тут ми задати не може, то стереотип *communicate*можна не вказувати (він неявно мається на увазі). Для прецеденту *зі складання комп'ютерів*не має значення який саме актор буде з ним взаємодіяти - *Інженер по збірці настільних комп'ютерів*або *Інженер по збірці ноутбуків.*Тому ми ввели ще одного актора - *Інженер по збірці*, з яким зв'язали перших двох акторів ставленням узагальнення ( *Generalization*).   

Відношення між прецедентами *Робота із замовленням*і *Управління інформацією про клієнта*- відношення розширення, оскільки коли актор *Менеджер по роботі з клієнтами*працює з замовленням (оформляє, змінює і т.д.), то не завжди при цьому він управляє інформацією про клієнтів. Відношення між прецедентами *зі складання комп'ютерів*і *Вимога необхідних комплектуючих*- відношення включення, оскільки для складання комп'ютерів обов'язково потрібно замовляти необхідні комплектуючі зі складу.

**2. Потік подій для прецедентів головної діаграми прецедентів**Потоки подій для прецедентів будемо описувати за наступним шаблоном:   

* Х.1 передумови;
* Х.2 головний потік;
* Х.3 під-потоки;
* Х.4 альтернативні потоки;
* Х.5 постумови.

де Х - число від одиниці до кількості прецедентів.  
  
**Потік подій для прецеденту «Робота з замовленням».**

**1.1 Передумови.**

Якщо замовлення оформляється для нового клієнта, то під-потік *додати нового клієнта (Add a New Client)*прецеденту *Управління інформацією про клієнта*повинен бути виконаний перед його початком.

**1.2 Головний потік.**

Прецедент починає виконуватися, коли менеджер підключається до системи і вводить своє ім'я і пароль. Система перевіряє правильність пароля (Е-1) і виводить можливі варіанти дій:

*додати (Add)*,

*змінити (Change),*

*видалити (Delete)*,

*переглянути (View)*

*вийти (Exit).*

Якщо обрана операція *додати (Add)*, S-1: виконується потік *додати нове замовлення (Add a New Order)*. Якщо обрана операція *змінити (Change)*, S-2: виконується потік *змінити замовлення (Change Order)*. Якщо обрана операція *видалити (Delete)*, S-3: виконується потік *видалити замовлення (Delete Order).*Якщо обрана операція *переглянути (View)*, S-4: виконується потік *переглянути замовлення (View Order).*Якщо обрана операція *вийти (Exit)*прецедент завершується.

**1.3 Під-потоки.**

S-1: *додати нове замовлення (Add a New Order)*Система відображає діалогове вікно, що містить поле, в якому менеджер повинен вибрати тип комп'ютера (настільний або ноутбук). Користувач вибирає необхідний тип. Система відображає поле для вибору клієнта і список можливих комплектуючих для обраного типу комп'ютера, в якому менеджер відзначає обрані клієнтом комплектуючі. Менеджер заповнює поля (E-2). Система запам'ятовує введені дані і роздруковує рахунок для оплати. Потім прецедент починається спочатку.

S-2: *змінити замовлення (Change Order)*Система відображає діалогове вікно, що містить список замовлень і поле для введення номера замовлення. Менеджер вибирає необхідний замовлення зі списку або вводить номер замовлення в поле (Е-3). Система відображає інформацію про даний замовленні. Менеджер робить необхідні зміни (Е-2). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

S-3: *видалити замовлення (Delete Order)*Система відображає діалогове вікно, що містить список замовлень і поле для введення номера замовлення. Менеджер вибирає необхідний замовлення зі списку або вводить номер замовлення в поле (Е-3). Система видаляє обраний замовлення (Е-4). Потім прецедент починається спочатку.

S-4: *переглянути замовлення (View Order)*Система відображає діалогове вікно, що містить список замовлень і поле для введення номера замовлення. Менеджер вибирає необхідний замовлення зі списку або вводить номер замовлення в поле (Е-3). Система відображає інформацію про обраний замовленні. Коли менеджер перегляне інформацію, прецедент почнеться спочатку.

**1.4 Альтернативні потоки**

Е-1: введено неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент.

Е-2: вибрані не всі комплектуючі, необхідні для складання комп'ютера або комплектуючих немає в наявності. Менеджер повинен змінити склад комп'ютера або завершити прецедент.

Е-3: введений неправильний номер замовлення. Менеджер повинен повторити введення або завершити прецедент.

Е-4: система не може видалити замовлення. Інформація зберігається, система видалить замовлення пізніше. Виконання прецеденту триває.

**Потік подій для прецеденту «Управління інформацією про клієнта».**

**2.1 Передумови.**

**2.2 Головний потік.**

Прецедент починає виконуватися, коли менеджер підключається до системи і вводить своє ім'я і пароль. Система перевіряє правильність пароля (Е-1) і виводить можливі варіанти дій: *додати (Add)*, *змінити (Change), видалити (Delete)*, *переглянути (View)*або *вийти (Exit).*Якщо обрана операція *додати (Add)*, S-1: виконується потік *додати нового клієнта (Add a New Client)*. Якщо обрана операція *змінити (Change)*, S-2: виконується потік *змінити дані про клієнта (Change Client Data)*. Якщо обрана операція *видалити (Delete)*, S-3: виконується потік *видалити клієнта (Delete Client).*Якщо обрана операція *переглянути (View)*, S-4: виконується потік *переглянути дані про клієнта (View Client Data).*Якщо обрана операція *вийти (Exit)*прецедент завершується.

**2.3 Під-потоки.**

S-1: *додати нового клієнта (Add a New Client)*Система відображає діалогове вікно, що містить поля для введення даних про нового клієнта. Користувач заповнює поля (Е-2). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

S-2: *змінити дані про клієнта (Change Client Data)*Система відображає діалогове вікно, що містить список клієнтів і поле для введення номера клієнта. Менеджер вибирає необхідного клієнта зі списку або вводить його номер в поле (Е-3). Система відображає інформацію про даному клієнті. Менеджер робить необхідні зміни (Е-2). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

S-3: *видалити клієнта (Delete Client)*Система відображає діалогове вікно, що містить список клієнтів і поле для введення номера клієнта. Менеджер вибирає необхідного клієнта зі списку або вводить його номер в поле (Е-2). Система видаляє вибраного клієнта (Е-4). Потім прецедент починається спочатку.

S-4: *переглянути дані про клієнта (View Client Data)*Система відображає діалогове вікно, що містить список клієнтів і поле для введення номера клієнта. Менеджер вибирає необхідного клієнта зі списку або вводить його номер в поле (Е-3). Система відображає інформацію про обраний клієнта. Коли менеджер перегляне інформацію, прецедент почнеться спочатку.

**2.4 Альтернативні потоки**

Е-1: введено неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент.

Е-2: заповнені не всі поля. Менеджер повинен заповнити незаповнені поля або завершити прецедент.

Е-3: введений неправильний номер клієнта. Менеджер повинен повторити введення або завершити прецедент.

Е-4: система не може видалити клієнта. Інформація зберігається, система видалить клієнта пізніше. Виконання прецеденту триває.  
          
      **Потік подій для прецеденту «Облік надходження та видачі комплектуючих.**

**3.1 Передумови.**

**3.2 Головний потік.**

Прецедент починає виконуватися, коли завскладом підключається до системи і вводить своє ім'я і пароль. Система перевіряє правильність пароля (Е-1) і виводить можливі варіанти дій: *додати (Add)*, *відзначити (Mark)*або *вийти (Exit).*Якщо обрана операція *додати (Add)*, S-1: виконується потік *внести пропозиції, що надійшли комплектуючі (Add a New Components)*. Якщо обрана операція *відзначити (Mark)*, S-2: виконується потік *зробити позначку про видачу комплектуючих (Mark Components)*. Якщо обрана операція *вийти (Exit)*прецедент завершується.

**3.3 Під-потоки.**

S-1: *внести пропозиції, що надійшли комплектуючі (Add a New Components)*Система відображає діалогове вікно, що містить поля для введення найменування комплектуючих, їх кількості, постачальника. Завскладом заповнює зазначені поля (Е-2). Система запам'ятовує введені дані. Потім прецедент починається спочатку. S-2: *зробити позначку про видачу комплектуючих (Change Order)*Система відображає список комплектуючих, що знаходяться на складі. Завскладом напроти потрібних комплектуючих вводить кількість виданих (Е-3). Система запам'ятовує введені дані. Потім прецедент починається спочатку.

**3.4 Альтернативні потоки**Е-1: введено неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент. Е-2: заповнені не всі поля. Користувач повинен заповнити пропущені поля або завершити прецедент. Е-3: зазначено кількість виданих комплектуючих, що перевищує їх кількість на складі. Користувач повинен повторити введення або завершити прецедент.

**Потік подій для прецеденту «Збірка комп'ютерів».**

**4.1 Передумови.**

**4.2 Головний потік.**

Прецедент починає виконуватися, коли інженер по збірці підключається до системи і вводить своє ім'я і пароль. Система перевіряє правильність пароля (Е-1) і виводить можливі варіанти дій: *переглянути (View)*, *відзначити (Mark)*або *вийти (Exit).*Якщо обрана операція *переглянути (View)*, S-1: виконується потік *Переглянути наряд на складання комп'ютера (View an Make Computer Order)*. Якщо обрана операція *відзначити (Mark)*, S-2: виконується потік *зробити позначку про статус зібраного комп'ютера за нарядом (Mark Computer)*. Якщо обрана операція *вийти (Exit)*прецедент завершується.

**4.3 Під-потоки.**

S-1: *Переглянути наряд на складання комп'ютера (View an Make Computer Order)*Система відображає діалогове вікно, що містить список нарядів і поле для введення номера наряду. Інженер вибирає необхідний наряд зі списку або вводить його номер в поле (Е-2). Система відображає інформацію про обраний вбранні. Коли інженер перегляне інформацію, прецедент почнеться спочатку. S-2: *зробити позначку про статус зібраного комп'ютера (Mark Computer)*Система відображає діалогове вікно, що містить список нарядів. Біля необхідного наряду інженер робить позначку про статус комп'ютера за даним нарядом. Інженер зберігає зміни. Потім прецедент починається спочатку.

**4.4 Альтернативні потоки**

Е-1: введено неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент. Е-2: заповнені не всі поля. Користувач повинен заповнити пропущені поля або завершити прецедент. Е-3: введений неправильний номер наряду. Інженер повинен повторити введення або завершити прецедент.  
          
**Потік подій для прецеденту «Вимога необхідних комплектуючих.**

**5.1 Передумови.**

**5.2 Головний потік.**

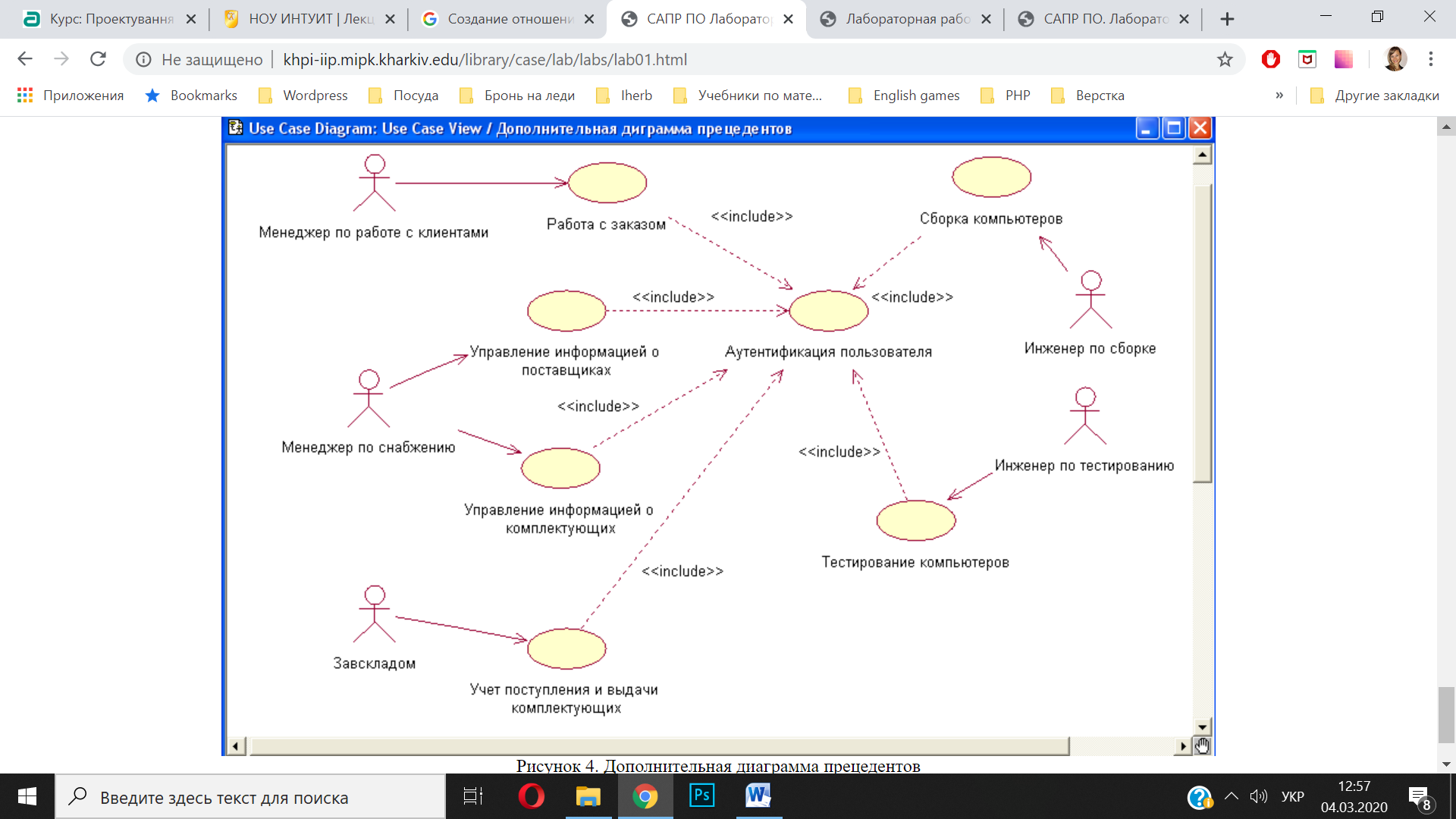
Прецедент починає виконуватися, коли інженер по збірці підключається до системи і вводить своє ім'я і пароль. Система перевіряє правильність пароля (Е-1) і виводить можливі варіанти дій: *переглянути (View)*, *зажадати (Order)*або *вийти (Exit).*Якщо обрана операція *переглянути (View)*, S-1: виконується потік *переглянути витребувані комплектуючі на складі (View Ordered Components on Warehouse)*. Якщо обрана операція *зажадати (Order)*, S-2: виконується потік *зажадати необхідні комплектуючі на складі (Order Required Components on Warehouse)*. Якщо обрана операція *вийти (Exit)*прецедент завершується.

**5.3 Під-потоки.**

S-1: *Переглянути витребувані комплектуючі на складі (View Ordered Components on Warehouse)*Система відображає наступну інформацію про всі зроблених замовленнях даними інженером по збірці: дата затребування, найменування комплектуючих, їх кількість, замовлення виконано чи ні. Коли інженер по збірці переглянув список, він повідомляє систему. Прецедент починається спочатку. S-2: *зажадати необхідні комплектуючі на складі (Order Required Components on Warehouse)*Система відображає діалогове вікно, що містить поля для введення списку необхідних комплектуючих і їх кількості. Інженер по збірці заповнює його. Система запам'ятовує введені дані. Потім прецедент починається спочатку.

**5.4 Альтернативні потоки**Е-1: введено неправильне ім'я або пароль. Користувач повинен повторити введення або завершити прецедент.  
             Опис потоків подій для прецедентів *Управління інформацією про постачальників*і *Управління інформацією про комплектуючих*аналогічно опису для прецеденту *Управління інформацією про клієнта;*для прецеденту *Тестування комп'ютерів*- прецеденту *зі складання комп'ютерів.*     

**3. Створення додаткової діаграми прецедентів.**  
Як видно з опису потоку подій для всіх прецедентів кожен з них включає перевірку користувача. Перевірка здійснюється одноманітно для будь-якого прецеденту. Тому її можна представити у вигляді окремого прецеденту *Аутентифікація користувача*, пов'язаного ставленням включення з усіма іншими. Результат створення діаграми показаний на рис. 4:

  
Рис 4. Додаткова діаграма прецедентів

**Завдання :**

1. створити головну діаграму прецедентів, задавши на ній варіанти використання і акторів;
2. додати відносини між акторами і варіантами використання;
3. створити додаткову діаграму прецедентів . Діаграма повинна включати в якості акторів всі типи даної суті: людина, інша система, подія.
4. додати опис акторів і варіантів використання;
5. для кожного варіанту використання задати потік подій у вигляді окремого файлу і прикріпити його до варіанту використання.

**Лабораторна робота №3**

**Розробка UML діаграми класів**

**Мета :**ознайомитися з системами розробки UML діаграм класів і навчитися застосовувати їх для проектування прикладних задач .

**Теоретичні відомості:**

Вид з точки зору проектування є основним етапом концептуальної опрацювання моделі. На даному етапі вводяться основні абстракції, визначаються класи і інтерфейси за допомогою яких реалізується рішення поставлених завдань. Якщо прецеденти тільки декларують поведінку системи, то на етапі розробки вигляду з точки зору проектування визначається якими засобами ці прецеденти будуть реалізовані. Статичні аспекти даного виду розробляються за допомогою діаграми класів, динамічні - за допомогою діаграм взаємодій і станів (автомат).

Діаграма класів містять класи, інтерфейси, кооперації, а так же зв'язку між ними. Починати розробку діаграми класів слід з визначення класів, що відповідають основним сутностей системи, які, як правило, визначаються на початкових етапах розробки при описі предметної області. Тут слід вирішити які сутності зручніше змоделювати як класи, а які як їх атрибути. Наприклад, якби була потрібна в рамках факультету вказати для кожної кафедри завідувача, краще сутність **завідувач кафедри**зробити атрибутом класу **кафедра**із зазначенням на клас **викладачі (**асоціація "один до одного" **),**а не вводити окремий клас **завідувач кафедри.**

При моделюванні необхідно пам'ятати, що для кожного класу повинна відповідати деяка реальна сутність або концептуальна абстракція з області, з якою має справу користувач або розробник. Добре структурований клас має такі властивості:

є чітко окресленою абстракцією деякого поняття зі словника проблемної області або області рішення;

містить невеликий, точно визначений набір обов'язків і виконує кожну з них;

підтримує чіткий поділ специфікацій абстракції і її реалізації;

зрозумілий і простий, але в той же час допускає розширення і адаптацію до нових завдань.

В рамках розробки моделі АРМ секретаря кафедри визначимо класи: **викладачі**, **студенти**, **аспіранти**, **дисципліни**, **групи**. Очевидно, що перші з них мають багато спільних атрибутів, тому введемо абстрактний клас **персона**, який буде инкапсулировать все властивості, що відносяться до людини в контексті розроблюваної системи (прізвище, ім'я, адреса і т.д.). В даному випадку **персона**буде суперкласом і зв'язуватися ставленням узагальнення з класами **викладачі**, **студенти**, **аспіранти.**

Атрибут **адреса**має власну структуру, щоб відобразити її можна ввести додатковий клас, назвемо його **T \_ ADR**(як прийнято в багатьох системах програмування назви класів починати з букви T ). Слід мати на увазі що, атрибут **адреса**класу **персона**є екземпляром класу **T \_ ADR**, тобто між цими класами встановлюється відношення залежності (відображається пунктирною стрілкою з відкритим наконечником, стрілка спрямована від залежного елемента до незалежного). У нашому випадку зміна структури класу **T \_ ADR**тягне за собою зміну класу **персона**через структуру відповідного атрибута ( **адреса**).

При моделюванні класу **T \_ ADR**атрибут **індекс**задамо за допомогою примітивного типу **T \_ POSTIDX**, що визначається як шестизначне десяткове число. Примітивні типи моделюються стереотипом " **type "**, діапазон значень вказується через обмеження, укладені у фігурні дужки.

У класі **викладач**виділимо специфічні атрибути, що стосуються лише викладачеві: **посаду**, **уч. ступінь**(вчений ступінь), **уч. звання (**вчене звання), **розряд**(розряд єдиної тарифної сітки). Атрибути **уч. ступінь**і **уч. звання**краще визначити спеціалізованими типами через перерахування. Перерахування моделюються класом зі стереотипом " **enum "**( enumeration - перерахування), допустимі значення записуються як атрибути, мітки, що визначають видимість атрибутів при цьому пригнічуються. У розглянутому прикладі через перерахування введемо спеціалізовані класи **T \_Должн**, **Т\_УчСт**, **Т\_УчЗв**, що визначають відповідно можливі посади, вчені ступені, вчені звання через перерахування. В даному випадку, як і всюди в подібних випадках при створенні класів, специфікує атрибути основного класу, встановлюються відносини залежності.

Для класу **студент**вводиться специфічний атрибут **номер заліковки**. Для класу аспірант визначаються специфічні атрибути **форма навчання**і **дата надходження**. Форма навчання визначаться спеціальним класом через перерахування **Т\_ФормОбуч**(очна, заочна).

Клас **група**має атрибути: **назва**, форма **навчання, число студ. (**Число студентів **).**З огляду на, що викладачі даної кафедри можуть вести заняття у груп з інших факультетів, вводиться додатковий клас **спеціальність**, з атрибутами **номер**(спеціальності), **назва**(спеціальності **), факультет**, типи яких в даній моделі не задані, хоча можуть визначатися через перерахування.

Клас **дисципліна**має атрибути: **номер**, **назва**, **цикл.**Атрибут **цикл**за допомогою введеного через перерахування спеціалізованого типу **Т\_Цікли**визначає до якого циклу відноситься дисципліна: до циклу гуманітарних та соціально-економічних дисциплін, математичних та природничих дисциплін, загально-професійних дисциплін, спеціальних дисциплін.

Атрибути **кількість годин**, **кількість семестрів**не можна вказати в класі **дисципліна**, оскільки вони залежать від спеціальності, тим більше не можна вказати їх в класі **спеціальність**. Дані атрибути відносяться до пари спеціальність-дисципліна і визначаються в класі - асоціації **Дисципліни-спеціальності**.



Рис. 1. Діаграма класів АРМ секретаря кафедри

**Прийоми роботи в Rational Rose для створення класів**

Для створення класів і відносин між ними в Rational Rose призначена панель *Toolbox,*яка розташована вертикально між браузером і вікном діаграми. На ній за замовчуванням представлені наступні кнопки: 

|  |  |
| --- | --- |
| **Назва кнопки** | **призначення кнопки** |
| Selection Tool | Перетворює курсор на стрілку покажчика для того, щоб можна було виділяти об'єкти |
| Text Box | Додавання до діаграми тексту |
| Note | Додавання до діаграми примітки |
| Anchor Note to Item | Зв'язування примітки з об'єктом на діаграмі |
| Class | Додавання на діаграму нового класу |
| Interface | Додавання на діаграму нового інтерфейсу |
| Unidirectional Association | Створення відносини асоціації |
| Association Class | Зв'язування класу асоціації зі ставленням асоціації |
| Package | Додавання на діаграму нового пакета |
| Dependency or instantiaties | Створення відносини залежності |
| Generalization | Створення відносини узагальнення |
| Realize | Створення відносини реалізації |

При бажанні склад кнопок на панелі *Toolbox*можна змінити, додавши на неї відсутні елементи. 

Для кожного класу можна задати ряд режимів відображення. Зробити це можна, клацнувши правою кнопкою миші по класу, і вибравши пункт меню *Options*(рис. 1):   

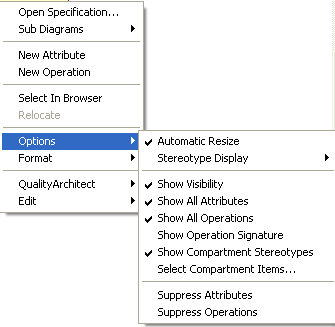


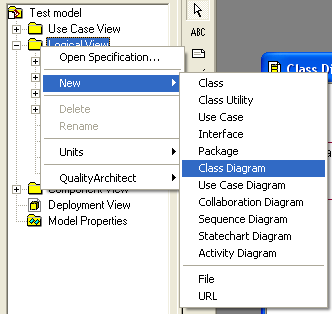
Рис 1. Контекстне меню класу

* *Automatic Resize*- автоматично підганяти розміри класу;
* *Stereotype Display*- спосіб відображення стереотипу класу :. Можливі варіанти: *none*(не відображати), *Label*(відображати в вигляді рядка над назвою класу), *Decoration*(відображати в вигляді графічного значка над назвою класу), *Icon*(відображати весь клас відповідним графічним значком);
* *Show Visibility*- відображати модифікатори доступу атрибутів і операцій;
* *Show All Attributes*- відображати всі атрибути класу;
* *Show All Operations*- відображати всі операції класу;
* *Show Operation Signature*- відображати сигнатуру операцій;
* *Suppress Attributes*- приховувати блок атрибутів;
* *Suppress Operations*- приховувати блок операцій.

Для завдання режимів відображень за замовчуванням для всіх класів передбачений пункт меню Tools -> Options, де у вкладці *Diagram*можна задати ці ж значення. 

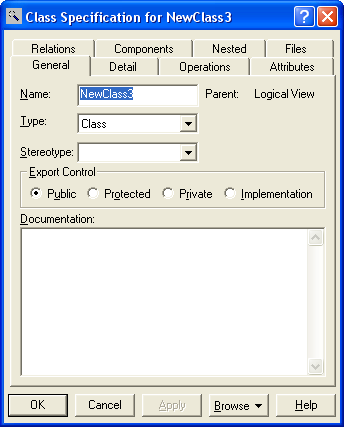
**Створення нової діаграми класів.**

Щоб створити діаграму класів необхідно натиснути правою кнопкою миші на логічному поданні браузера (Logical View). У меню, вибрати пункт New> Class Diagram (Створити> Діаграма Класів) (рис. 2). Далі ввести назву діаграми, після чого двічі клацнути по ній в браузері, щоб відкрити її.

  
Рис 2. Створення діаграми класів

**Створення нового класу.**

Для створення нового класу потрібно клацнути по кнопці *Class*на панелі *Toolbox*і потім по вільному місцю вікна діаграми, після чого можна ввести назву класу. Після створення класу потрібно визначити його властивості. Для цього потрібно двічі по ньому клацнути або ж викликати для нього контекстне меню і вибрати пункт *Open Specification ...,*після чого відкриється вікно специфікації класу, що містить ряд вкладок (рис. 3) *.*     

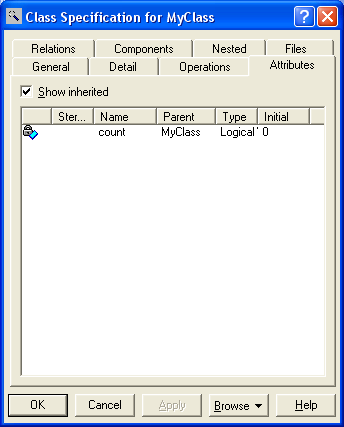
  
Рис 3. Вікно специфікації класу

Розглянемо вміст кожної вкладки:

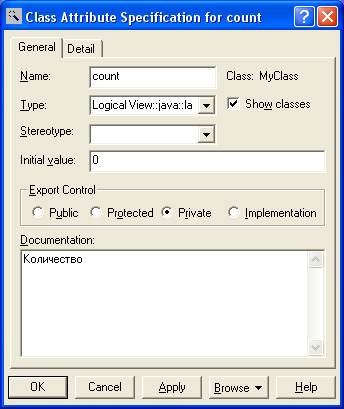
* *General*- тут задаються загальні властивості класу: ім'я класу (Name), тип класу (Type), стереотип (Stereotype), видимість класу за межами його пакета (Export Control) і текстовий опис класу (Documentation);
* *Detail*- задаються деякі подробиці класу: потужність (Multiplicity), кількість пам'яті, необхідне при виконанні об'єктів даного класу (Space), час життя примірників класу (Persistence), скільки дій може виконуватися в один часовий інтервал (Concurrency), чи є клас абстрактним ( Abstract), форРисьні параметри для певних типів класів (Formal Arguments);
* *Operations*- задаються операції класу;
* *Attributes*- задаються атрибути класу;
* *Relations -*відображаються відносини класу з іншими класами;
* *Components -*відображаються пов'язані з класом компоненти;
* *Nested -*додавання вкладених класів;
* *Files -*додавання файлів, що містять додаткову інформацію про клас.

**Додавання нових атрибутів до класу.**

Для додавання нового атрибуту до класу необхідно відкрити специфікацію класу і вибрати вкладку *Attributes*(рис. 4) *.* 

   
Рис 4. Вкладка Attributes вікна специфікації класу

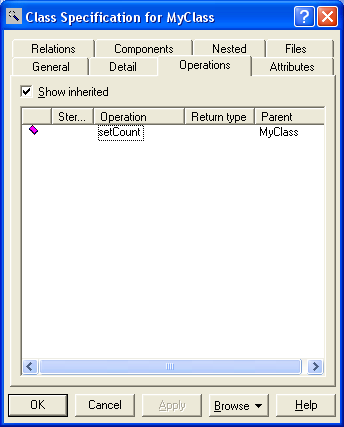
Далі слід натиснути правою кнопкою по вільному місцю основного поля вкладки, вибрати пункт *Insert*і ввести ім'я атрибута. Для завдання властивостей атрибута слід двічі клацнути по ньому або ж викликати для нього контекстне меню і вибрати пункт *Specification ...*, після чого відкриється вікно специфікації атрибута, що містить дві вкладки (рис. 5):   

  
Рис 5. Вікно специфікації атрибута класу

* *General -*задаються загальні властивості атрибута: ім'я (Name), тип (Type), стереотип (Stereotype), початкове значення (Initial value), видимість атрибута за межами його класу (Export Control) і текстовий опис класу (Documentation);
* *Detail*- задаються деякі подробиці атрибута: спосіб зберігання атрибута в класі (Containment), чи є атрибут статичним (Static), похідним, тобто створеним на основі інших атрибутів (Derived).

**Додавання нових операцій до класу.**

Щоб додати новий операції до класу необхідно відкрити специфікацію класу і вибрати вкладку *Operations*(рис. 6):

  
Рис 6. Вкладка Operations вікна специфікації класу

Далі слід натиснути правою кнопкою по вільному місцю основного поля вкладки, вибрати пункт *Insert*і ввести ім'я операції. Для завдання властивостей операції слід двічі клацнути по ній або ж викликати для неї контекстне меню і вибрати пункт *Specification ...*, після чого відкриється вікно специфікації операції, що містить наступні вкладки (рис. 7):   

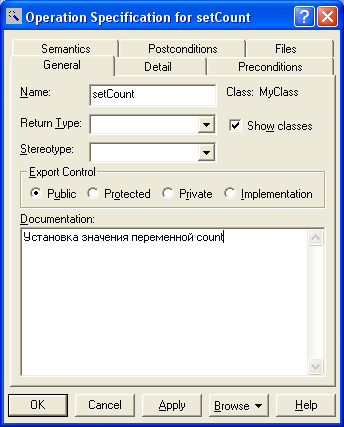


Рис 7. Вікно специфікації операції класу

* *General -*задаються загальні властивості атрибута: ім'я (Name), тип значення (Return Type), показувати чи все класи пакета (Show classes), стереотип (Stereotype), видимість операції за межами її класу (Export Control) і текстовий опис операції ( Documentation);
* *Detail*- задаються деякі подробиці операції: параметри (Arguments); список операцій, які може виконувати клієнт класу, і порядок їх виконання (Protocol); уточнення операції, пов'язані з конкретним мовою програмування (Qualification); виняткові ситуації (Exceptions); передбачуваний обсяг, який буде потрібно під час виконання операції (Size); передбачуваний час виконання операції (Time); поведінка операції при наявності декількох потоків управління (Concurrency);
* *Preconditions*- задаються передумови, тобто умови, які повинні бути виконані перед запуском операції;
* *Postconditions*- задаються постумови, тобто умови, які повинні бути виконані після завершення роботи операції;
* *Semantics*- дозволяє описати, що буде робити операція (логіка операції);
* *Files -*додавання файлів, що містять додаткову інформацію про клас.

**Додавання параметрів до операції класу.**

Для додавання параметрів до операції класу необхідно відкрити специфікацію операції, вибрати вкладку *Details*, в ній натиснути правою кнопкою миші по полю *Arguments*і ввести ім'я параметра. Для завдання властивостей параметра слід двічі клацнути по ньому або ж викликати для нього контекстне меню і вибрати пункт *Specification ...*, після чого відкриється вікно специфікації параметра, що містить наступну вкладку (рис.8):      

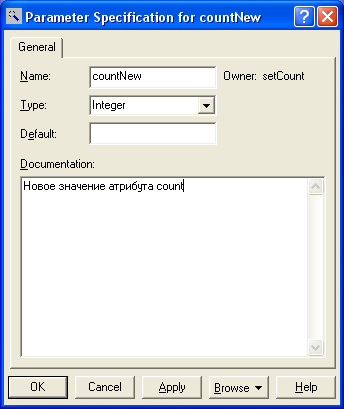


Рис 8. Вікно специфікації параметрів операції класу

* *General -*тут задаються ім'я параметра (Name), тип (Type), значення за замовчуванням (Default) і текстовий опис параметра (Documentation).

**Створення відносин між класами.**

*Загальне зауваження.*Для будь-якого типу відносин завдання його властивостей здійснюється однаковим способом - або двічі клацнути по лінії, або викликати для неї контекстне меню і вибрати пункт *Specification ....*Частина властивостей відносини можна задати прямо в його контекстному меню. Спільними властивостями для всіх типів відносин є: ім'я (Name), стереотип (Stereotype) і текстовий опис відносини (Documentation). Нижче будуть описані тільки специфічні для кожного типу відносин властивості.     
  
**Залежності.**  
Є найбільш загальною формою відносини в мові UML. Всі інші типи відносин можна вважати окремим випадком даного відносини. Ставлення залежності показує, що зміна одного класу призводить до зміни іншого класу. Найчастіше застосовується, коли один клас використовує інший як аргумент. Зображується пунктирною лінією зі стрілкою, спрямованої від залежного класу до незалежного.  
Для створення відносини залежності слід вибрати кнопку *Dependency or instantiaties*на панелі *Toolbox,*потім клацнути мишкою по залежному класу і не відпускаючи кнопки миші перетягнути стрілку на незалежний клас. Для даного типу відносини можна задати наступні властивості: чи буде мати залежний клас доступ до не-public елементів незалежного класу (Friendship required), видимість відносини (Export Control), початкову (Multiplicity from) і кінцеву потужність (Multiplicity) відносини. 

**Асоціації, агрегації і композиції.**

Оскільки відносини асоціації, агрегації і композиції відрізняються дуже незначно, то розглянемо роботу з ними в загальному. *Ставлення асоціації*показує, що один клас якимось чином пов'язаний з іншим класом (аналог зв'язку в діаграмі «Сутність-Зв'язок»). Зображується суцільною лінією, що з'єднує класи. *Ставлення агрегації -*окремий випадок асоціації. Являє собою відношення типу «ціле / частина». Зображується у вигляді простої асоціації з незакрашенним ромбом з боку «цілого». Агрегація не є спадкуванням, оскільки всі класи-«частини» в агрегації є цілком самостійними, зі своїми атрибутами і операціями, що відрізняються від атрибутів і операцій класу-«ціле». *Ставлення композиції -*окремий випадок агрегації. Служить для виділення спеціальної форми відносини «ціле-частина», при якій складові частини в деякому сенсі перебувають всередині цілого. Специфіка взаємозв'язку між ними полягає в тому, що частини не можуть виступати у відриві від цілого, тобто зі знищенням цілого знищуються і всі його складові частини. Зображується у вигляді суцільної лінії з зафарбовані ромбом біля класу «ціле». Для створення одного з наведених вище відносин слід вибрати кнопку *Unidirectional Association*на панелі *Toolbox,*потім клацнути мишкою по одному класу і не відпускаючи кнопки миші перетягнути стрілку на інший клас. За замовчуванням створюється однонаправлена асоціація, тобто клас, від якого йде стрілка, знає про загальні атрибути та операціях другого класу, але не навпаки. Для створення двобічної асоціації слід викликати для відносини контекстне меню і вибрати пункт *Navigable.*Для даних типів відносин можна задати наступні властивості:       
  
  
  
   
  
  
  
     

* вкладка *General -*роль класів в даному відношенні (Role A і Role B);
* вкладка *Detail -*чи є ставлення похідним (Derived); клас-асоціація, пов'язаний з даним відношенням (Link Element); напрямок ролі (Name Direction); обмеження - деяка умова, яка повинна виконуватися (Constraints);
* вкладки *Role A General, Role B General -*задаються загальні властивості ролей класів, що беруть участь в відношенні: назва ролі (Role), видимість ролі (Export Control) і текстовий опис ролі (Documentation);
* вкладки *Role A Detail, Role B Detail -*задаються деякі подробиці ролей: ім'я ролі (Role), обмеження для ролі (Constraints), потужність ролі (Multiplicity), напрямок відносини (Navigable), чи є ставлення агрегацией (Aggregate), чи є атрибути класів, які беруть участь в відношенні, статичними (Static); чи є класи відносини дружніми (Friend); яким чином будуть включатися створені атрибути агрегації - за значенням або за посиланням (Containment of Class) (при виборі значення By Value тип відносини зміниться на композицію), додавання кваліфікаторів (Keys / Qualifiers).

**Узагальнення.**

Це відношення між загальною сутністю (суперкласом, або батьком) і її конкретним втіленням (субклассов, або нащадком). Узагальнення іноді називають відносинами типу "є", маючи на увазі, що одна сутність є приватним вираженням інший, більш загальної. Узагальнення означає, що об'єкти класу-нащадка можуть використовуватися всюди, де зустрічаються об'єкти класу-батька, але не навпаки. Зображується у вигляді лінії з великою незафарбовані стрілкою.  
Для створення відносини узагальнення слід вибрати кнопку *Generalization*на панелі *Toolbox,*потім клацнути мишкою по класу-нащадка і не відпускаючи кнопки миші перетягнути стрілку на клас-батько. Для даного типу відносини можна задати наступні властивості: чи буде мати клас-нащадок доступ до не-public елементам класу-батька (Friendship required), видимість відносини (Export Control), чи буде успадковуватися тільки одна копія класу-батька (Virtual inheritance).          
  
**Реалізації.**  
Це відношення між двома елементами моделі, при якому один елемент (клієнт) реалізує поведінку, задане іншим (постачальником). Зображується у вигляді пунктирною лінії з великою незафарбовані стрілкою, що вказує на постачальника. Найчастіше реалізації використовують для визначення відносин між інтерфейсом і класом або компонентом, який надає оголошені в інтерфейсі операції або послуги.  
Для створення відносини узагальнення слід вибрати кнопку *Realize*на панелі *Toolbox,*потім клацнути мишкою по об'єкту-клієнта і не відпускаючи кнопки миші перетягнути стрілку на об'єкт-постачальник. Специфічних властивостей у даного типу відносин не передбачено.  

**завдання:**

1. Вивчити приклад розробки UML - діаграми класів, викладений в теоретичних відомостях.
2. Визначити сутність проектованої інформаційної системи і їх атрибути.
3. Розробити діаграму класів засобами Rational Rose .

**Лабораторна робота № 4 .**

**Розробка UML діаграм послідовностей і кооперацій .**

**Мета :**ознайомитися UML діаграмами послідовностей і кооперації і навчитися їх розробляти засобами Rational Rose .

**Теоретичні відомості:**

Розглянемо процес створення нового запису про студента засобами діаграми послідовності.

Створення нового запису передбачає права адміністратора, тому дійовою особою цього взаємодії буде адміністратор ( **адмін**). Даний об'єкт буде відразу був введений на діаграмі прецедентів, тому перетягнемо його на діаграму послідовності з браузера виду з точки зору прецедентів.

Слід звернути увагу, що на діаграмах взаємодії фігурують об'єкти, тобто конкретні екземпляри класів (ім'я об'єкта завжди підкреслюється).

Ведемо об'єкти: **форма введення**, **менеджер записів**, запис про студента **Петрова**(як конкретний приклад записи про студента), **менеджер транзакцій**. Даний набір об'єктів є типовим при модифікації записи в таблиці бази даних.

**Форма введення**- елемент призначеного для користувача інтерфейсу, є типову форму введення даних про студента (прізвище, ім'я, по батькові, адреса і т.д.). У нашому випадку є кілька доопределении конкретну реалізацію стандартного інтерфейсу **редагування**класу **електронна таблиця.**Оскільки спеціально інтерфейс редагування даних про студента нами не вводився на діаграмі класів, тому явно вказувати клас для об'єкта **форма введення**не будемо.

**Менеджер записів**- об'єкт, що володіє стандартним набором можливостей з управління даними при роботі з електронною таблицею. Даний набір можливостей успадковується класом **студенти**від класу **електронна таблиця**. Для об'єкта **Менеджер записів**явно вказується клас, екземпляром якого він є - **студенти**.

**Петров**- конкретний запис про студента Петрова, новий елемент таблиці про студентів. Тут явно вкажемо введений клас **запис про студента**. Подібні об'єкти зазвичай існують тимчасово для посилки відповідної інформації в базу даних при транзакціях. Після закінчення транзакції даний об'єкт може бути знищений. Відповідний записи об'єкт може бути створений знову при необхідності редагування інформації.

**Менеджер транзакцій**- об'єкт, що забезпечує виконання закінченої операції над базою даних, в даному випадку створення нового запису про студента Петрова. На даний об'єкт покладається виконання також ряду системних функцій, які супроводжують трансакцію. Прикладом менеджерів трансакцій є, наприклад, BDE (використовуються для доступу з додатків Delphi до баз даних Paradox , Dbase і ін.), ADO (використовується для доступу до баз MS Access з різних додатків).

Діаграма послідовності введення нового запису про студента в АРМ секретаря кафедри представлена ​​на рис. 7.



Рис. 7. Введення даних про студента. Діаграма послідовності.

На діаграмі послідовності визначимо передачу повідомлень між об'єктами: **створити новий запис**(транслюється від об'єкта до об'єкта до кінця ланцюжка як повідомлення **зберегти запис**); **відкрити форму**(до форми введення); **ввести П.І.Б., адреса. (**Введення даних по студенту), далі ці дані транслюються повідомленнями **зберегти П.І.Б., адреса.**Від **менеджера транзакцій**передається повідомлення зібрати **інформацію про студента**, що забезпечує зворотний зв'язок з базою даних, і нарешті рефлексивне повідомлення **менеджера транзакцій**пойменоване як **зберегти запис в БД**, забезпечує закінчення транзакції.

При бажанні можна дана взаємодія уявити діаграмою кооперації, що ілюструє насамперед структурний аспект взаємодії (рис.8). Дану діаграму можна побудувати з попередньої в автоматичному режимі (в Rational Rose натисканням клавіші F 5).



Рис. 8. Введення даних про студента. Діаграма кооперації.

У разі потреби проект можна доповнити й іншими діаграмами взаємодії, що розкривають роботу прецедентів.

**Прийоми роботи в Rational Rose для створення об'єктів і повідомлень**

Для створення об'єктів і повідомлень між ними в Rational Rose призначена панель *Toolbox,*яка розташована вертикально між браузером і вікном діаграми. На ній за замовчуванням представлені наступні кнопки:  

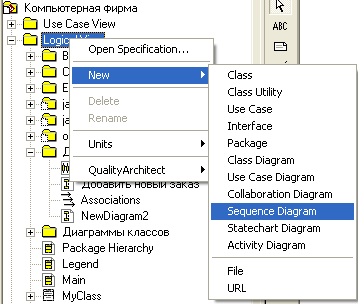
|  |  |
| --- | --- |
| **Назва кнопки** | **призначення кнопки** |
| Selection Tool | Перетворює курсор на стрілку покажчика для того, щоб можна було виділяти об'єкти |
| Text Box | Додавання до діаграми тексту |
| Note | Додавання до діаграми примітки |
| Anchor Note to Item | Зв'язування примітки з об'єктом на діаграмі |
| Object | Додавання на діаграму нового об'єкта |
| Object Message | Додавання нового сполучення між об'єктами |
| Message to Self | Створення рефлексивного повідомлення самому собі |
| Return Message | Створення відносини повернення |
| Destruction Marker | Створення маркера знищення |

При бажанні склад кнопок на панелі *Toolbox*можна змінити, додавши на неї відсутні елементи. 

Для відображення номера повідомлення в Rational Rose необхідно в меню Tools> Options> вкладка Diagram поставити галочку біля напису Sequence numbering.

**Створення нової діаграми послідовності.**

Щоб створити діаграму послідовності необхідно натиснути правою кнопкою миші на представленні Варіантів Використання браузера (Use Case View) або Логічному поданні (Logical View). У меню, вибрати пункт New> Sequence Diagram (Створити> Діаграма послідовності) (рис. 1). Далі ввести назву діаграми, після чого двічі клацнути по ній в браузері, щоб відкрити її.

  
Рис 1. Створення діаграми послідовності

**Додавання нового об'єкта.**

Для створення нового об'єкта потрібно клацнути по кнопці *Object*на панелі *Toolbox*і потім по вільному місцю вікна діаграми. Далі необхідно зайти в специфікацію об'єкта, двічі клацнувши мишкою по межі прямокутника або правою кнопкою миші по ньому і вибравши пункт меню *Open Specification ...*(рис. 2). 

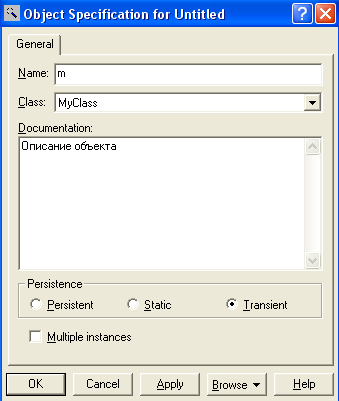


Рис 2. Вікно специфікації об'єкта

У вікні специфікації для об'єкта можна задати: ім'я об'єкта (Name); клас, екземпляром якого є цей об'єкт (Class), якщо класу ще немає в проекті, то тут його можна створити, вибравши в списку, що випадає значення *<New>*; текстовий опис (Documentation); час життя об'єкта (Persistence); чи є цей об'єкт мультіоб'ектом (Multiple instances). Об'єкт на діаграму послідовності можна додати і іншим способом - шляхом перестасківанія відповідного класу з браузера на діаграму. При цьому автоматично буде створено анонімний об'єкт - екземпляр цього класу. 

**Додавання нового повідомлення.**

Для додавання нового повідомлення між об'єктами У вікні специфікації для об'єкта можна задати: ім'я об'єкта (Name); клас, екземпляром якого є цей об'єкт (Class), якщо класу ще немає в проекті, то тут його можна створити, вибравши в списку, що випадає значення *<New>*; текстовий опис (Documentation) У вікні специфікації для об'єкта можна задати: ім'я об'єкта (Name); клас, екземпляром якого є цей об'єкт (Class), якщо класу ще немає в проекті, то тут його можна створити, вибравши в списку, що випадає значення *<New>*; текстовий опис (Documentation) У вікні специфікації для об'єкта можна задати: ім'я об'єкта (Name); клас, екземпляром якого є цей об'єкт (Class), якщо В вікні специфікації для об'єкта можна задати: ім'я об'єкта (Name); клас, екземпляром якого є цей об'єкт (Class), якщо класу ще немає в проекті, то тут його можна створити, вибравши в списку, що випадає значення *<New>*; текстове *опис (Documentation)*класу ще немає в проекті, то тут його можна створити, вибравши в списку, що випадає значення *<New>*; текстовий опис (Documentation) У вікні специфікації для об'єкта можна задати: ім'я об'єкта (Name); клас, екземпляром якого є цей об'єкт (Class), якщо класу ще немає в проекті, то тут його можна створити, вибравши в списку, що випадає значення *<New>*; текстовий опис (Documentation) необхідно клацнути по кнопці *Object Message*або *Return Message*на панелі *Toolbox*, клацнути по лінії життя об'єкта-відправника повідомлення (клієнт) і не відпускаючи кнопку перетягнути стрілку на лінію життя об'єкта-одержувача (сервер).         

Для додавання нового повідомлення об'єкта самому собі необхідно клацнути по кнопці *Message to Self*на панелі *Toolbox*і клацнути по лінії життя об'єкта. 

Після створення повідомлення можна визначити його властивості. Для цього потрібно двічі по ньому клацнути або ж викликати для нього контекстне меню і вибрати пункт *Open Specification ...,*після чого відкриється вікно специфікації повідомлення, що містить дві вкладки (рис. 3) *.*  

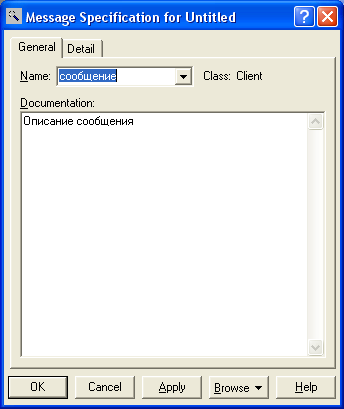
**

Рис 3. Вікно специфікації повідомлення

Вміст кожної вкладки:

* *General*- задаються ім'я повідомлення (Name) і текстовий опис повідомлення (Documentation);
* *Detail*- задаються синхронізація посилаються повідомлень (Synchronization) і частота повідомлення (Frequency).

Синхронізація посилаються повідомлень може приймати одне з наступних семи значень:

* *Simple (просте) -*використовується за умовчанням. Чи означає, що всі повідомлення виконуються в одному потоці управління.
* *Synchronous (синхронне)*- застосовується, коли клієнт посилає повідомлення і чекає відповіді на нього від сервера.
* *Balking (з відмовою стає в чергу)*- клієнт посилає повідомлення сервера. Якщо сервер не може негайно прийняти повідомлення, воно скасовується.
* *Timeout (з лімітованим часом очікування)*- клієнт посилає повідомлення сервера, а потім чекає вказаний час. Якщо протягом цього часу сервер не приймає повідомлення, воно скасовується.
* *Procedure Call*- повідомлення, що вимагає виклику операції або процедури об'єкта-одержувача.
* *Asynchronous (асинхронне) -*клієнт посилає повідомлення сервера і продовжує свою роботу, не очікування підтвердження про отримання.
* *Return -*повідомлення повернення з виклику операції або процедури.

Частота повідомлень може приймати одне з таких двох значень:

* *Aperiodic (апериодическое)*- повідомлення відправляється нерегулярно. Воно може бути відправлено тільки один раз або кілька разів, але через різні проміжки часу.
* *Periodic (періодичне)*- означає, що повідомлення регулярно надсилається через певні проміжки часу.

**Приклад виконання роботи.**

Створювати діаграми взаємодії будемо для сценарію "Додати нове замовлення" прецеденту "Робота з замовленням". У цьому сценарії крім основного потоку існують ще й альтернативні потоки. Хоча стандарт мови UML допускає розгалуження на діаграмах послідовності і кооперації, ми, щоб не захаращувати наші діаграми, обмежимося розглядом тільки випадку, коли користувач правильно вводить свій пароль, правильно заповнює необхідні поля і введені дані без помилок зберігаються в базі даних. У разі необхідності альтернативні потоки можна показати на додаткових діаграмах послідовності і кооперації.

Діаграми взаємодії будемо створювати в Логічному поданні браузера. Для того, щоб відокремити ці діаграми від інших (які ми вже створили або створимо в подальшому), створимо спочатку новий пакет в Логічному поданні - *Діаграми взаємодії*, в якому будуть розташовуватися створені далі діаграми.

Побудова будь-якої діаграми взаємодії починається з визначення переліку об'єктів, які будуть брати участь у взаємодії. Для обраного сценарію в лабораторній роботі № 2 була розроблена діаграма класів. Примірники класів цієї діаграми і будуть учасниками діаграм взаємодії.

*Примітка:*Rational Rose дозволяє, маючи одну з двох типів діаграм взаємодії, створити другу. Для цього необхідно відкрити наявну діаграму взаємодії і вибрати пункт меню Browse> Create Sequence (Collaboration) Diagram. Автоматично буде створена діаграма другого типу з таким же ім'ям, в тому ж пакеті і з таким же вмістом, що і перша. Єдиний недолік цього прийому - в створеній діаграмі елементи не будуть автоматично вирівнюватися. Тому якщо вихідна діаграма досить велика, то у створеній діаграмі складно буде розібратися, тому що елементи можуть налазить один на одного.

У даній роботі ми обидва типи діаграм взаємодії будемо будувати з нуля.

**Створення діаграми послідовності для сценарію "Додати нове замовлення" прецеденту "Робота з замовленням"**

Для створення діаграми послідовності необхідно натиснути правою кнопкою миші по пакету Діаграми взаємодії і в меню вибрати пункт New> Sequence Diagram, ввести її ім'я, після чого двічі клацнути по ній в браузері, щоб відкрити її (рис. 1).

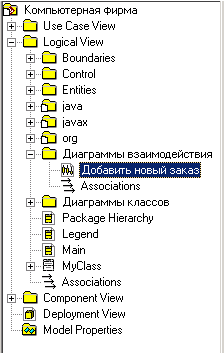


Рис 1. Створення діаграми послідовності

Побудова діаграми послідовності починається з розміщення на ній об'єктів, які будуть обмінюватися повідомленнями. Спочатку необхідно розмістити об'єкти, які посилають повідомлення, а потім об'єкти, які отримують їх. Ініціатором взаємодії виступає актор *Менеджер по роботі з клієнтами.*Тому на діаграмі він буде знаходиться в лівому кутку. Далі розміщуємо (рис. 2):

- об'єкт класу *OrderOptions*( *Параметри роботи із замовленням*) *,*що відповідає за вибір можливого дії із замовленням в розглянутому прецедент; 

- об'єкт класу *AddNewOrder*( *Додавання нового замовлення*), що відповідає за додавання замовлення;

- об'єкт класу *OrderManager*( *Менеджер по роботі з замовленнями*), що відповідає за обробку потоку подій розглянутого прецеденту;

- об'єкт класу *Order*( *Замовлення*);

- об'єкт класу *Client*( *Клієнт*);

- об'єкт класу *ComponentPart*( *Комплектуючий виріб*).

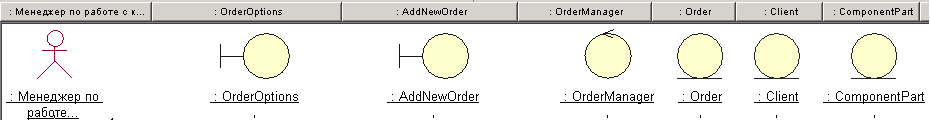


Рис 2. Розподіл об'єктів на діаграмі послідовності

Тепер на діаграмі слід розмістити повідомлення, якими будуть обмінюватися об'єкти (рис. 3):

|  |  |  |  |
| --- | --- | --- | --- |
| **номер повідомлення** | **Об'єкт - відправник повідомлення** | **Об'єкт - одержувач повідомлення** | **Назва** |
| 1 | Менеджер по роботі з клієнтами | OrderOptions | введення пароля |
| 2 | OrderOptions | OrderOptions | перевірка пароля |
| 3 | Менеджер по роботі з клієнтами | OrderOptions | вибір операції "додати" |
| 4 | OrderOptions | AddNewOrder | відображення полів введення |
| 5 | Менеджер по роботі з клієнтами | AddNewOrder | вибір типу комп'ютера |
| 6 | AddNewOrder | OrderManager | отримання списку клієнтів |
| 7 | OrderManager | Client | отримання списку клієнтів |
| 8 | Client | AddNewOrder | список клієнтів |
| 9 | AddNewOrder | AddNewOrder | відображення списку клієнтів |
| 10 | Менеджер по роботі з клієнтами | AddNewOrder | вибір клієнта |
| 11 | AddNewOrder | OrderManager | отримання списку комплектуючих |
| 12 | OrderManager | ComponentPart | отримання списку комплектуючих |
| 13 | ComponentPart | AddNewOrder | список комплектуючих |
| 14 | AddNewOrder | AddNewOrder | відображення списку комплектуючих |
| 15 | Менеджер по роботі з клієнтами | AddNewOrder | \* Вибір необхідних комплектуючих |
| 16 | Менеджер по роботі з клієнтами | AddNewOrder | зберегти замовлення |
| 17 | AddNewOrder | OrderManager | передача управління |
| 18 | OrderManager | Order | зберегти |

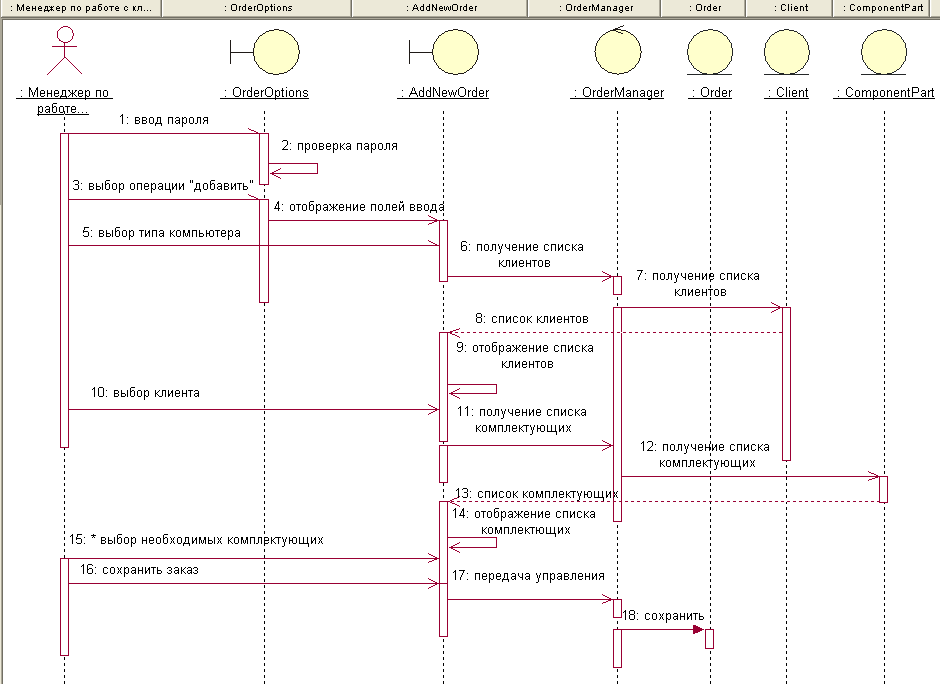


Рис 3. Підсумкова діаграма послідовності

Для відображення номера повідомлення в Rational Rose необхідно в меню Tools> Options> вкладка Diagram поставити галочку біля напису Sequence numbering.

**Створення діаграми кооперації для сценарію "Додати нове замовлення" прецеденту "Робота з замовленням"**

Для створення діаграми кооперації необхідно натиснути правою кнопкою миші по пакету Діаграми взаємодії і в меню вибрати пункт New> Collaboration Diagram, ввести її ім'я, після чого двічі клацнути по ній в браузері, щоб відкрити її (рис. 4).

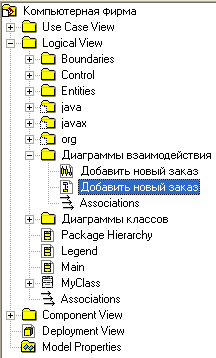
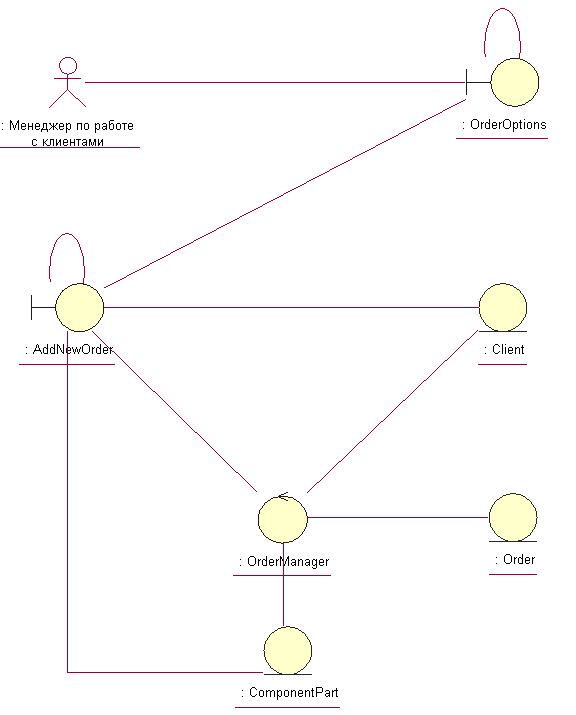


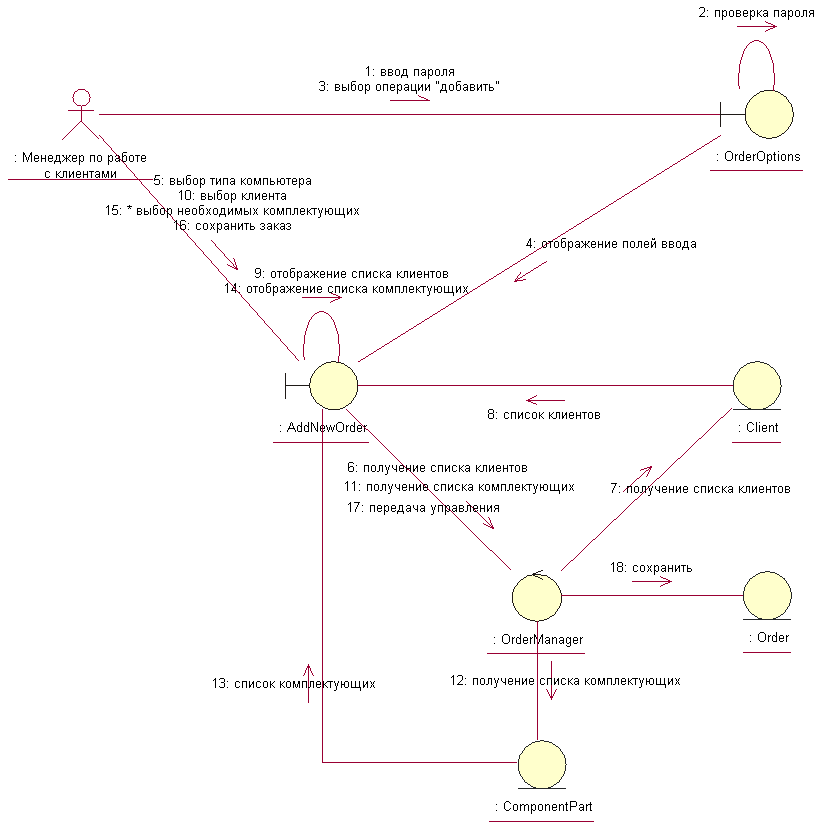
Рис 4. Створення діаграми кооперації

Побудова діаграми кооперації починається з розміщення на ній об'єктів, які будуть обмінюватися повідомленнями. Перелік об'єктів на даній діаграмі такий же, як і на попередній. Далі необхідно додати на діаграму зв'язку між об'єктами, які обмінюються повідомленнями (рис. 5):

* *Менеджер по роботі з клієнтами*і *AddNewOrder*
* *Менеджер по роботі з клієнтами*і *OrderOptions*
* *AddNewOrder*і *OrderOptions*
* *AddNewOrder*і *OrderManager*
* *AddNewOrder*і *Client*
* *AddNewOrder*і *ComponentPart*
* *OrderManager*і *Client*
* *OrderManager*і *ComponentPart*
* *OrderManager*і *Order*

  
Рис 5. Об'єкти і зв'язку на діаграмі кооперації

Останніми на діаграму кооперації додаються сполучення між об'єктами. Перелік повідомлень між об'єктами такої ж, як і на діаграмі послідовності (рис. 6).

  
Рис 6. Підсумкова діаграма кооперації

**завдання:**

1. Вивчити приклад розробки UML - діаграми послідовностей, викладений в теоретичних відомостях.
2. Виявити взаємодіючі об'єкти.
3. Розробити діаграми послідовностей засобами Rational Rose для проектованої системи.

**Лабораторна робота № 5**

**Розробка UML діаграм діяльності**

**Мета :**ознайомитися з UML діаграмами діяльності і навчитися їх розробляти засобами Rational Rose .

**Теоретичні відомості:**

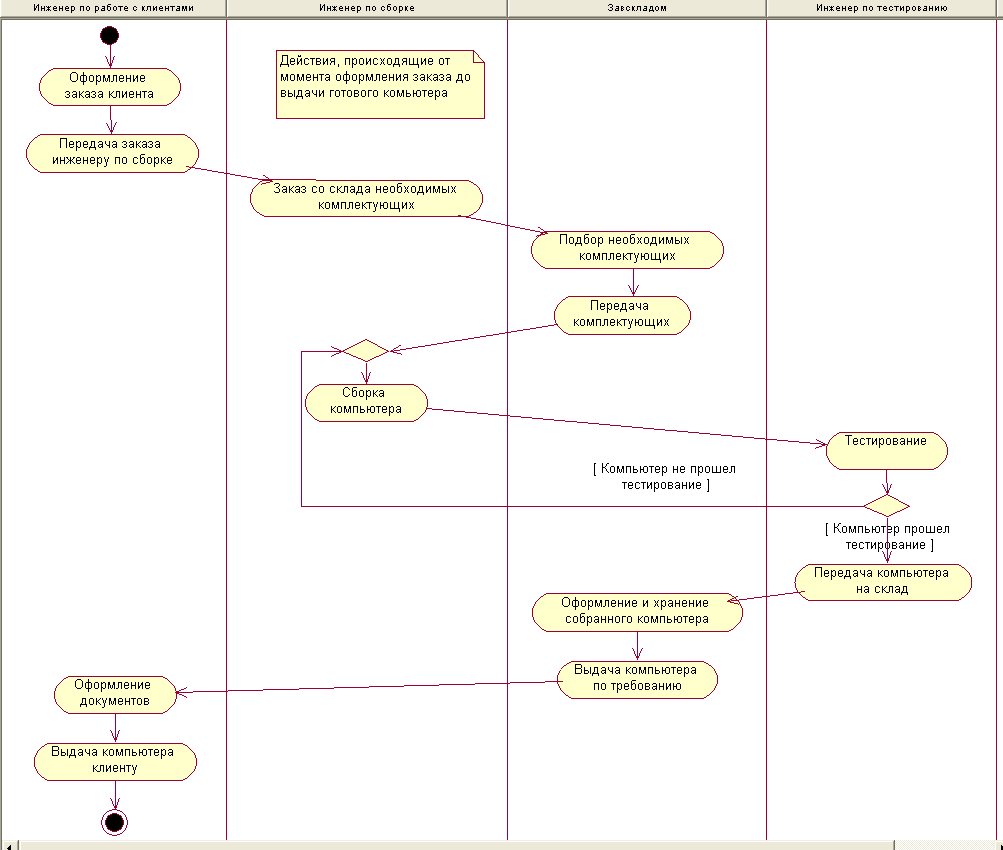
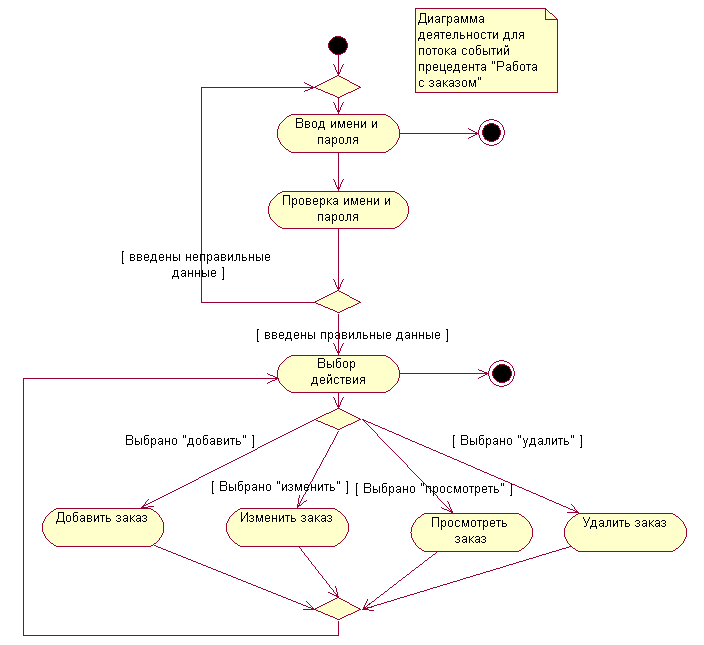
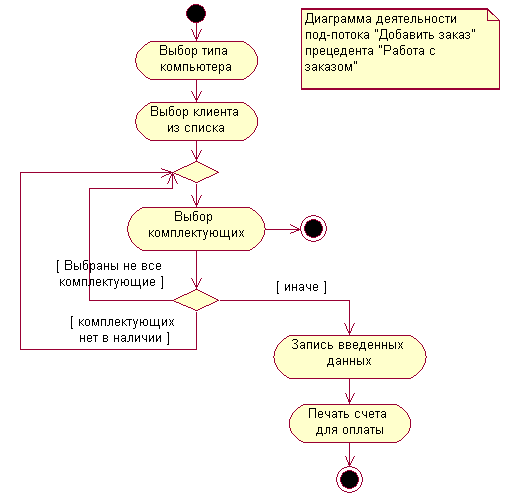
**1. Створення діаграми діяльності для бізнес-процесу підприємства зі складання комп'ютерів.**  
Розглянемо в цілому, що відбувається на підприємстві від моменту оформлення замовлення на збірку комп'ютера до видачі готового комп'ютера. Після оформлення замовлення менеджер по роботі з клієнтами передає його менеджеру по збірці, який перш ніж розпочати складання замовляє необхідні комплектуючі зі складу. На складі завідувач підбирає необхідні комплектуючі (в разі їх відсутності замовляє їх у менеджера з постачання) і передає їх інженеру по збірці. Після отримання комплектуючих менеджер по збірці здійснює збірку комп'ютера і передає його інженеру по тестуванню. Якщо комп'ютер не пройшов тестування, він повертається для повторного складання. При успішному завершенні тестування комп'ютер передається на склад на зберігання. Зі складу комп'ютер на вимогу передається інженеру по роботі з клієнтами, який оформляє на нього документи і видає клієнтові. Для створення діаграми дій необхідно натиснути правою кнопкою миші по Поданню Варіантів Використання і в меню вибрати пункт New> Activity Diagram, ввести її ім'я, після чого двічі клацнути по ній в браузері, щоб відкрити її. Результат побудови діаграми показаний на рис. 1:     
  


Рис 1. Діаграма діяльності бізнес-процесу

**2. Створення діаграми діяльності потоку події варіанти використання "Робота з замовленням"**Потік подій варіанту використання "Робота з замовленням" складається з головне потоку, під-потоків і альтернативних потоків. Щоб не захаращувати діаграму покажемо потік подій на декількох діаграмах діяльності. На першій з них (умовно назвемо її головною) покажемо дії для основного потоку і пов'язаний з ним альтернативний потік (рис. 2). Під-потоки можна буде показати шляхом декомпозиції відповідного дії головної діаграми.

  
Рис 2. Діаграма діяльності для потоку подій прецеденту "Робота з замовленням"

Для декомпозиції дії діаграми діяльності слід клацнути по ній правою кнопкою миші і в меню вибрати пункт Sub Diagrams> New Activity Diagram. Приклад декомпозиції дії На рис. 3 показана діаграма діяльності для під-потоку "Додати замовлення", яка є декомпозицією дії "Додати замовлення" головною діаграми діяльності. 

  
Рис 3. Діаграма діяльності для дії "Додати замовлення"

**завдання:**

1. зі здати діаграму діяльності, що описує один з бізнес-процесів обраної предметної області;
2. створити діаграму діяльності, що описує потік подій одного з варіантів використання, створеного в лабораторній роботі № 2 .
3. в звіт включити створені діаграми діяльності із зазначенням того, який бізнес-процес і потік подій якого варіанту використання вони описують.

**Лабораторна робота № 6 .**

**Розробка UML діаграм станів .**

**Мета :**ознайомитися з UML діаграмами станів і навчитися їх разрабат и вать засобами Rational Rose .

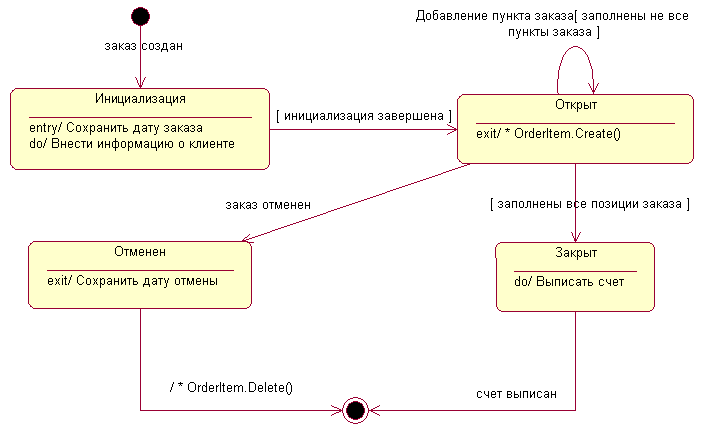
**Теоретичні відомості:**

**Приклад виконання роботи.**

Діаграми станів застосовуються, як правило, для моделювання поведінки класів, прецедентів або системи в цілому.  
Складемо діаграму станів для класу *Order (Замовлення)*, оскільки в нашій моделі він найбільш часто змінюватиме свій стан. Замовлення може знаходиться в декількох стану:   

* при створенні замовлення він переходить в стан *Ініціалізація,*в якому виконуються деякі попередні дії;
* після завершення ініціалізації замовлення переходить в стан *Відкрито*, в якому до замовлення додаються нові пункти. Вихід з цього стану можливий або в разі скасування замовлення, або в разі заповнення всіх необхідних пунктів замовлення;
* якщо заповнені всі необхідні пункти замовлення, то він переходить в стан *Закрито*, в якому відбувається виписка рахунку. Вихід з цього стану можливе лише після того, як рахунок буде виписаний;
* якщо замовлення скасований, то зі стану *Відкрито*він переходить в стан *Скасовано*. При виході з цього стану відбувається видалення всіх пунктів замовлення.

Діаграма станів для класу *Order*представлена на рис.1: 

  
Рис 1. Діаграма станів для класу Order

Першим станом на діаграмі станів є початковий стан. При виконанні події "замовлення створений" замовлення переходить в стан *Ініціалізація.*При вході в цей стан виконується вхідний дію "Зберегти дату замовлення". Основна дія, яке буде виконуватися протягом усього часу, поки замовлення буде знаходиться в цьому стані, це "Внести інформацію про клієнта". Перехід з цього стану в стан *Відкрито*відбудеться тільки при виконанні сторожового умови "ініціалізація завершена". 

У стані *Відкрито*є вихідна дія і перехід в себе. Перехід в себе означає, що подія ініціює перехід, відбувається вихід з поточного стану, виконується певна дія, після чого відбувається повернення в початковий стан. Оскільки при переході в себе відбувається вихід зі стану і повторний вхід в нього ж, то виконується дія, асоційоване з переходом, і, крім того, дія при вході в стан. У стані *Відкрито*до замовлення додаються нові пункти, причому їх можна додати лише в тому випадку, якщо є незаповнені пункти. Для показу цього ми використовували перехід в себе "Додавання пункту замовлення" зі сторожовим умовою "заповнені не всі пункти замовлення". Вихід з цього стану відбудеться в двох випадках - або коли виконається сторожову умову "заповнені всі позиції замовлення" (при цьому замовлення перейде в стан *Закрито*), або коли настане подія "замовлення скасований" (при цьому замовлення перейде в стан *Скасовано*). При виході з стану виконається дія виходу "\* OrderItem.Create ()" (створення пункту замовлення). Символ "\*" вказує на те, що ця дія виконається багато разів (по числу доданих пунктів в замовлення).

У стані *Закрито*присутній тільки внутрішня дія - "Виписати рахунок". В цей стан замовлення переходить зі стану *Відкрито*тільки при виконанні сторожового умови "заповнені всі позиції замовлення". Вихід з цього стану і перехід в кінцеве відбудеться при настанні події "рахунок виписаний". У стан *Скасовано*замовлення переходить зі стану *Відкрито*при настанні події "замовлення скасований". При виході з нього виконується дія виходу "Зберегти дату скасування". При переході з цього стану в кінцеве виконується дія "\* OderItem.Delete ()" (видалення пункту замовлення). Тут також варто "\*", оскільки ця дія буде виконуватися багато разів.     
  
 

**Завдання:**разробити діаграму станів розроблених класів або прецедентів проектованої інформаційної системи .