Проектування сучасних програмних систем є досить складним і дорогим процесом. Як і проектування інших систем процес проектування включає кілька етапів.

1. **Етап технічного завдання**. На цьому етапі замовник або розроблювач у тісному зв'язку із замовником формулюють основні вимоги до розроблюваного програмного продукту. Даний етап є дуже важливим, так як якщо параметри програмного продукту будуть сформульовані невірно, то вся інша розробка буде під загрозою. Крім того, ці вимоги важливі для процесів верифікації й валідації.

2. **Етап ескізного проекту**. Відбувається укрупнене пророблення одного або декількох варіантів програмного продукту з метою вибору оптимального варіанта системи, який би відповідав закладеним на попередньому етапі вимогам. У випадку, якщо доведено, що неможливо задовольнити вимоги замовника, то або відбувається коректування технічного завдання, або припиняється розробка системи.

3. **Етап проектування**. Проводиться проектування програмного продукту.

4. **Етап тестування**.

5. **Етап приймання програмного продукту замовником**.

Усі етапи проектування програмних систем повинні супроводжуватися документацією, яка відповідає вимогам відповідних державних, галузевих або корпоративних стандартів.

Мова UML (Unified Modeling Language) є мовою загального призначення для розробки в галузі програмного забезпечення, яка призначена для забезпечення стандартного способу візуалізації дизайну системи.

При моделюванні складних систем використовується принцип абстрагування, тобто включення в модель найголовніших властивостей системи, які використовуються для виконання системою своїх функцій.

Крім даного принципу також використовуються принципи багатомодельності і ієрархічної побудови моделей складних систем. Принцип багатомодельності означає, що для повної моделі складної системи потрібне не одне, а деяка кількість представлень, кожне з яких описує з свого боку поведінку і структуру системи.

Принцип ієрархічної побудови моделей складних систем означає, що спочатку треба побудувати так звану первісну модель на найвищому рівні абстрагування. Далі, будуються наступні моделі з більшою деталізацією.

Для групування елементів моделі у мові UML використовується пакет. Пакети (підпакети) можуть бути вкладеними в інші пакети. Пакети на діаграмі зображуються у вигляді прямокутника з невеликим прямокутником, приєднаним до лівої частини верхньої сторони першого.

Крім імені пакета можуть використовуватися так звані стереотипи - певні ключові слова мови UML, наприклад, framework, toplevel та ін.

Відношення вкладеності пакетів можна зобразити розмістивши один пакет - прямокутник у інший.

Модель можна визначити як підклас пакету, яка є абстракцією системи і призначена для досягнення конкретних цілей. Цілі зазвичай формулюються як вихідні вимоги до проектованої системи і у мові UML зображуються як варіанти використання.

За допомогою мови UML модель системи представляється у графічному вигляді, так званій діаграмі. Діаграма це графічне зображення, на якому представлені елементи моделі у вигляді графу.

Існують такі канонічні діаграми:

* варіантів використання (прецедентів) (use case diagram)
* класів (class diagram)
* кооперації (collaboration diagram)
* послідовності (sequence diagram)
* станів (statechart diagram)
* діяльності (activity diagram)
* компонентів (component diagram)
* розгортання (deployment diagram)

За допомогою сукупності цих діаграм можна повністю описати складну систему.

Так як діаграма варіантів використання представляє найбільш загальну модель системи, вона є початковою у сукупності канонічних діаграм.

У діаграмах UML використовується не тільки графічна, а і текстова інформація у вигляді стереотипів, позначених значень і обмежень.

Стереотип - текст у кутових лапках, який розширює семантику метамоделі. Є визначені стереотипи, які є ключовими словами мови UML і стереотипи користувача.

Позначене значення (tagged value) це представлення властивості у вигляді пари "ім'я – значення".

Синтаксис позначеного значення: {тег = значення}.

Обмеження (constraint) — це деякий логічний вираз (умова), яка обмежує значення певного елемента розроблюваної моделі .

Обмеження найчастіше визначаються самими розробниками. Синтаксис обмеження - текст у фігурних дужках. Існує спеціальна мова об'єктних обмежень (Object Constraint Language, OCL), яка заснована на UML і призначена для формального запису обмежень.

**Лабораторна робота 1.**

**Діаграма прецедентів (use case diagram).**

У мові **UML** діаграми прецедентів дозволяють візуалізувати поведінку системи, підсистеми або класу, щоб користувачі могли зрозуміти як їх використовувати, а розроблювачі - реалізувати відповідний елемент.

На діаграмі прецедентів представлена сукупність акторів, які взаємодіють із системою через варіанти використання. Актор (діюча особа) це така сутність, яка ззовні взаємодіє із системою. В якості акторів можуть виступати люди, програми, пристрої або інші системи. Що стосується варіантів використання, то вони представляють сервіси системи для акторів.

Діаграми прецедентів звичайно містять у собі:

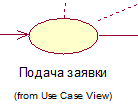
* прецеденти;
* ектори (актори);
* відносини залежності, узагальнення й асоціації.

Як і всі інші діаграми, вони можуть містити примітки й обмеження.

Іноді в діаграми прецедентів поміщають пакети, застосовувані для групування елементів моделі в більші блоки, а в ряді випадків і екземпляри прецедентів, особливо якщо треба візуалізувати конкретну систему, що виконується.

За варіантами використання стоять певні дії, які має зробити система взаємодіючи із актором. На діаграмі використання часто використовуються примітки або сценарії, які пояснюють логіку її компонентів.

Варіант використання зображується на діаграмі у формі еліпса, у якому є ім'я (дієслово) із словами пояснення (рис. 1).



**Рис. 1.** Зображення варіанта використання на діаграмі

Крім опису взаємодії між системою і користувачами, діаграма використання описує реакцію системи на повідомлення від користувачів. Також у варіантах використання можуть бути описи деталей реалізації сервісів і виняткових ситуацій. Інколи певна множина використання може представлятися як окремий пакет.

Актор зображується на діаграмі у вигляді "чоловічка", а знизу розміщується ім'я (рис. 2). По суті актор є певна сутність, яка у взаємодії із системою досягає своїх конкретних цілей.

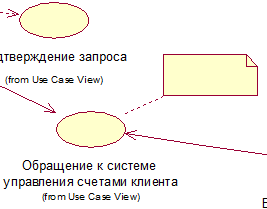


**Рис. 2.** Графічне позначення актора

У якості прикладу актора можна навести користувача системи із своїм логіном і паролем.

Прикладами приміток можуть слугувати коментарі розроблювача, обмеження і таке інше.

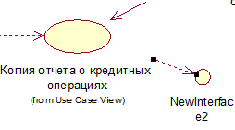
Примітки на діаграмі зображуються як прямокутник, у якому верхній правий кут загнутий (рис. 3). Примітка може стосуватись будь-якого елемента діаграми і з'єднується із ним пунктирною лінією. Пунктирних ліній можу бути декілька, якщо примітка стосується кількох елементів. Текст примітки вміщується усередині прямокутника.



**Рис. 3.** Позначення примітки на діаграмі

Елемент UML діаграми інтерфейс (interface) використовується для визначення параметрів моделі без представлення їх внутрішньої організації. Конкретно у діаграмах використання за допомогою інтерфейсів описують набір операцій, які реалізують сукупність сервісів для акторів. Атрибути, стани і спрямовані асоціації не використовуються у інтерфейсах.

Зображення інтерфейсу на діаграмі являє собою невелике коло з ім'ям біля нього (рис. 4). Іменем може бути або іменник, або текст ("мікрофон", "датчик", "пульт керування" та ін.). Імена на англійській мові прийнято починати з великої літери І: ISensor, IMonitor.

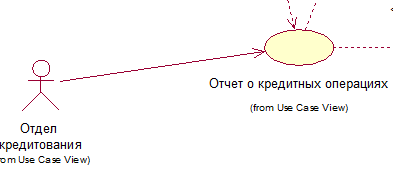
/

**Рис. 4.** Позначення інтерфейсу на діаграмі

Мова UML має такі види відносин між акторами й варіантами використання:

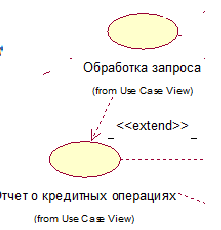
* Відношення асоціації (association relationship)
* Відношення розширення (extend relationship)
* Відношення узагальнення (generalization relationship)
* Відношення включення (include relationship).

Відношення асоціації визначає роль, яку відіграє актор по відношенню до варіанта використання. На діаграмі відношення асоціації зображується як суцільна лінія, яка зв'язує актора і варіант використання. До цієї лінії можуть використовуватися ім'я та кратність (рис. 5).



**Рис. 5.** Відношення асоціації на діаграмі.

Відношення розширення між варіантами використання зображується як пунктирна лінія із ключовим словом "extend" від того варіанта використання, який є розширенням для вихідного варіанта використання (рис. 6. ).



**Рис. 6.** Відношення розширення між варіантами використання на діаграмі.

Відношення розширення показує, що деякий варіант використання додає до власної поведінки поведінку іншого варіанта використання. Відношення розширення має певну умову і покажчик на точку розширення у базовому варіанті використання. Розширення спрацює у цій точці тільки якщо виконується дана умова.

Відношення узагальнення показує, що варіант використання - нащадок може бути узагальнений до варіанту використання - предка. При цьому нащадок успадковує всі властивості і поведінку свого предка і може мати свої власні властивості і варіанти поведінки. На діаграмі відношення узагальнення зображується як суцільна лінія із не зафарбованою стрілкою, що спрямована на варіант використання - батька (стрілка "узагальнення") (рис. 7).

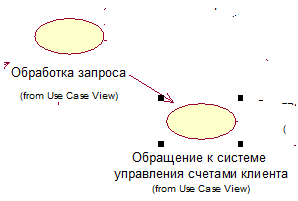
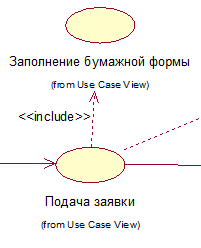


Рис. 7. Позначення відношення узагальнення між варіантами використання.

Відношення включення означає, що в поведінку одного варіанта використання включається поведінка іншого варіанта використання.

Варіант використання може включати в себе декілька варіантів і бути включеним у інші кілька варіантів.

На діаграмі використання відношення включення зображується у вигляді пунктирної лінії зі стрілкою і ключовим словом "include", що вказує на варіант, який включається (рис. 8. ).



**Рис. 8.** Позначення відношення включення .

**Лабораторна робота 2.**

**Діаграма станів (statechart diagram).**

Діаграма станів показує порядок зміни станів екземпляра класу (об'єкта), які спричинені іншими об'єктами або ззовні. Таким чином діаграма класів призначена для опису послідовності станів і переходів, які визначають поведінку об'єкта на протязі його життєвого циклу.

Крім об'єктів у діаграмах станів можуть бути специфіковані такі компоненти моделі як операції, методи, актори, підсистеми та варіанти використання.

В термінах теорії автоматів діаграма станів є графом, який представляє деякий автомат. В якості вершин даного графа виступають стани або псевдостани, а в якості дуг - переходи між станами. Діаграми станів можуть бути вкладені в інші діаграми станів.

Побудувати діаграму станів у середовищі IBM Rational Rose можна різними способами:

* У панелі інструментів натиснути на кнопку з позначенням діаграми станів - вказати бажаний тип і представлення діаграми - нова діаграма станів.
* У браузері проекту вибрати такі команди: **Logical View або Use Case View** - у контекстному меню : **New Statechart Diagram**.
* У браузері проекту розкрити **Logical View - виділити клас, операцію, пакет чи Use Case View (вибравши варіант використання)** і у контексному меню вибрати **New Statechart Diagram**.
* У головному меню вибрати команду **Browse State Machine Diagram** - представлення й тип діаграми.

У початковий момент часу об'єкт перебуває у так званому початковому стані, який не має ніяких внутрішніх дій (псевдостан). На діаграмі початковий стан зображується як зафарбоване коло, з якого стрілки переходів можуть тільки виходити.

У кінцевий момент часу об'єкт перебуває у так званому кінцевому стані, який не має ніяких внутрішніх дій (псевдостан). На діаграмі кінцевий стан зображується як зафарбоване коло, розміщене у інше коло, в який стрілки переходів можуть тільки входити (рис 9).

а) б)

**Рис. 9.** Початковий (а) і кінцевий (б) стан

В мові UML, використовуються наступні зарезервовані мітки:

* entry — вхідна дія;
* exit — вихідна дія;
* do — позначення діяльності, що має місце на протязі перебування об'єкта в даному стані.
* include — звертання до підавтомата.

В загальному випадку мітка дії позначає подію, що призводить до виконання відповідного виразу дії. Дані події є внутрішніми переходами.

Графічно для позначення переходу використовується суцільна лінія зі стрілкою, що вказує на цільовий стан . Поряд з переходом можна розміщувати текст у такому форматі:

<сигнатура події>'['<сторожова умова>']' < вираз дії>.

В свою чергу сигнатура події описується таким чином:

<ім'я події>'('<список параметрів, розділених комами>')'.

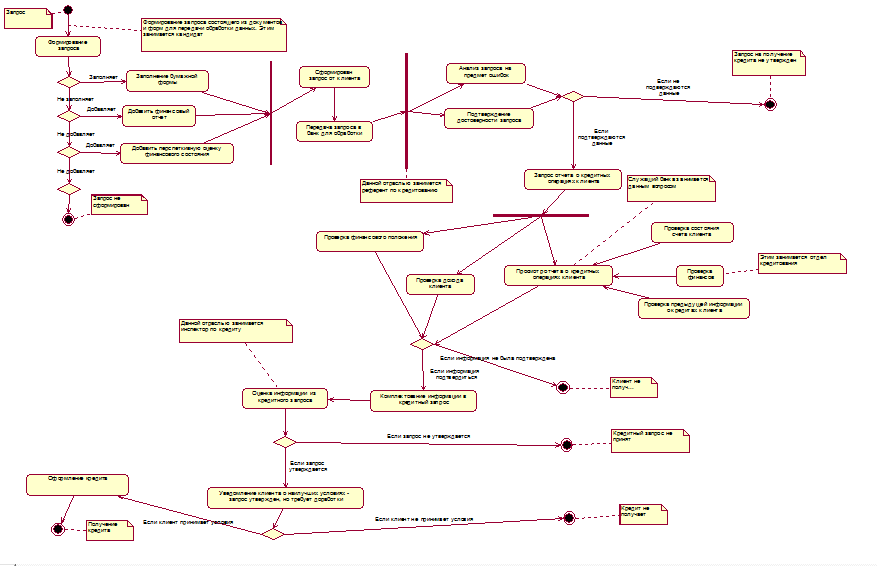
Під подією розуміється специфікація деякого факту, який відбувається у певному місці і у певний час. Події можуть бути упорядковані у часі і повернутися до попередніх подій не можна, якщо це явно не описано в моделі.

Якщо ніякого рядка тексту поряд зі стрілкою переходу немає, то діяльність, після якої спрацьовує даний перехід, повинна бути зрозуміла із контексту самої діаграми. Список параметрів події-тригера може бути відсутнім.

 Сторожова умова (guard condition) - це деякий бульовий вираз, він також може бути відсутнім. Сторожова умова обчислюється тільки після якщо відбувається відповідна подія-тригер. Якщо сторожова умова істинна, то відбувається даний перехід, якщо хибна - перехід спрацює.

З однією подією-тригером може бути пов'язано декілька переходів. Звичайно, декілька сторожових умов не повинні одночасно бути істинними. Коли настає подія-тригер, сторожові умови обчислюються заново.

На рис 10. наведений приклад діаграми станів для варіанта лабораторної роботи 1.



**Лабораторна робота 3.**

**Діаграма діяльності (activity diagram).**

*Діаграма діяльності* (Activity diagram) показує потік переходів від однієї діяльності до іншої. Діяльність (**Activity**) - це триваючий у часі неатомарний крок обчислень в автоматі. В остаточному підсумку діяльність представляє собою певну дію (**Action**), яка складена з атомарних обчислень. Графічно діаграма діяльності представляється у вигляді графа, що має вершини й ребра.

Як правило, діаграма діяльності має наступні складові частини:

* стани діяльності й стани дії;
* переходи;
* Q об'єкти.

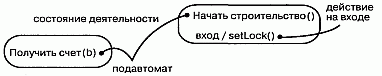
На діаграмі діяльності, яка моделює потік керування, зображуються різні події. Це, наприклад, обчислення виразів, створення або знищення об'єктів і таке інше. Такі атомарні обчислення відомі як стани дії і позначаються прямокутниками із закругленими краями (рис. 11). Всередині прямокутника дозволяється розміщувати вирази.



**Рис. 11** Стани дії

Стани дії не можуть бути декомпозовані. Крім того, вони атомарні і тривають час, яким можна знехтувати.

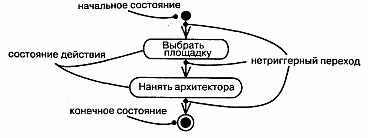
На противагу цьому стани діяльності можуть бути піддані подальшої декомпозиції і представленню у вигляді інших діаграм діяльності. Стани діяльності не атомарні, не перериваються і тривають значний час. Стан дії визначається як стан, який не може бути підданий декомпозиції. Стан діяльності - це стан, потік керування якого стладається із інших станів і дій. Гляньте більш пильно на внутрішню структуру стану діяльності, і ви знайдете ще одну діаграму діяльності. Як видно з рис. 12, стани діяльності й дій позначаються однаково, з тою відмінністю, що в перший може мати дії входу і виходу.



**Рис.12** Стани діяльності

Якщо треба зобразити складну дію, яка складається з деяких простих дій, то використовується стан піддіяльності (subactivity state). Він зображається як піктограма у правому нижньому куті символу стану дії.

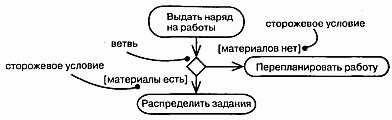
При завершенні дії чи діяльності стану, потік керування передається до подальшого стану. Такий перехід показується за допомогою переходів (**Transitions**). В **UML** перехід представляється звичайною лінією зі стрілкою, як показано на рис.13.



**Рис. 13** Нетригерні переходи

Оскільки керування після закінчення виконання у вихідному стані зразу передається далі, то подібні переходи є переходами по завершенню (не-тригерні: **Triggerless**). Далі, після дії вихідного стану, іде дія виходу, якщо така існує. Потім відбувається перехід потоку керування в наступний стан дії або діяльності і т. д. Керування може у такий спосіб переходити із стану у стан невизначено довго (у випадку діяльності, що незавершилася) або до попадания в кінцевий стан (нетригерні переходи можуть мати сторожові умови, які обумовлюють їхню активізацію)

Прості послідовні переходи зустрічаються найбільше часто, але їх одних недостатньо для моделювання будь-якого потоку керування. Як і в блок-схемі, для розгалудження шляхів виконання можна використовувати булеві вирази. На рис. 14 показаний приклад точки розгалудження (ромб). На діграмі діяльності розгалуження представляється у вигляді ромбу без тексту. Вхідна стрілка може бути тільки одна, вихідних - може бути декілька. Вхідна стрілка зазвичай входить до верхньої або лівої вершини ромба. Для кожної вихідної стрілки явно вказується відповідна сторожова умова у формі булевого виразу. Сторожові умови повинні відповідати таким вимогам: вони не повинні бути істинними одночасно для двох або більше вихідних переходів і вони повинні розповсюджуватись на всі можливі варіанти.

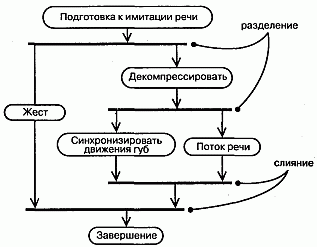


**Рис. 14** Розгалуження

Для переходу, який спрацьовує коли умови інших переходів хибні, можна застосовувати ключове слово **else.**

Крім простих і розгалуджених послідовних переходів у діграмах діяльності використовуються і паралельні потоки, як правило, для моделювання бізнес-процесів. В **UML** для позначення поділу й злиття таких паралельних потоків виконання використовується синхронізаційна риска, яка проводиться у вигляді жирної вертикальної або горизонтальної лінії. Кожний з потоків керування, що паралельно виконуються, існує в контексті незалежного активного об'єкта, який, як правило, моделюється або процесом, або обчислювальним потоком (рис.15) .

Після точки поділу діяльності, асоційовані з кожним шляхом у графі, продовжують виконуватися паралельно.

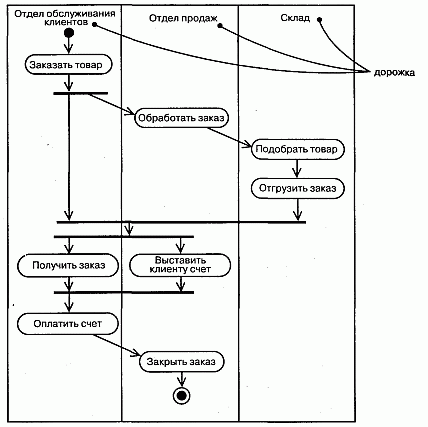


**Рис. 15** Поділ і злиття

Точка злиття являє собою механізм синхронізації декількох паралельних потоків виконання. У цю точку входять два або більше переходів, а виходить рівно один. Вище точки злиття діяльності, асоційовані із шляхами які прийшли до неї, виконуються паралельно. У точці злиття відбувається синхронізація паралельних потоків, тобто кожний з них чекає, поки всі інші досягнуть цієї точки, після чого виконання триває в рамках одного потоку.

Діаграми діяльності можуть моделювати не тільки алгоритми обчислень і потоки керування, а і бізнес-процеси.

Якщо моделюється бізнес-процес, можна для кожного відділу компанії створити свою групу станів діяльності, які відділяються одна від одної вертикальними лініями, так званими доріжками (**Swimlanes**) (див. рис. 16).



**Рис. 16** Доріжки

Кожній присутній на діаграмі доріжці привласнюється унікальне ім'я. Ніякої глибокої семантики доріжка не несе, хіба що може відбивати деяку сутність реального світу. Кожна доріжка представляє сферу відповідальності за частину всієї роботи, зображеної на діаграмі, і в остаточному підсумку може бути реалізована одним або декількома класами. На діаграмі діяльності, розбитої на доріжки, кожна діяльність належить рівно однієї доріжці, але переходи можуть перетинати границі доріжок.

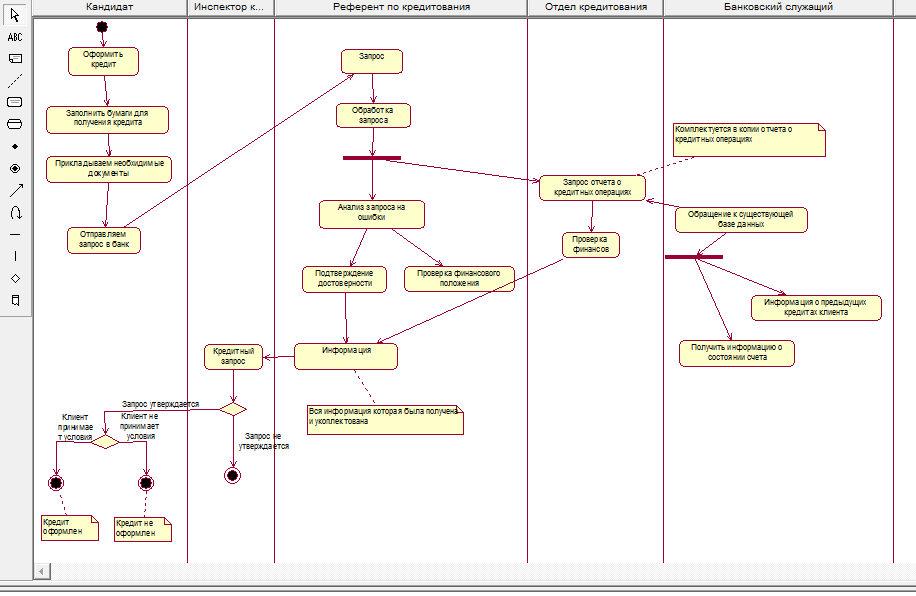
На діаграмі діяльності іноді треба зобразити об'єкти, так як вони виконують деякі дії або окреслюють певний результат дій.

Так як і класи об'єкти на діаграмі представляються у вигляді прямокутника, але при цьому ім'я об'єкта підкреслюється. Характеристика стану об'єкта може вказуватися у прямих дужках після імені об'єкта. Для приєднання об'єктів до станів використовується відношення залежності - пунктирна лінія зі стрілкою.

Інколи має значення розташування об'єкта на діаграмі. Якщо об'єкт знаходиться на межі двох доріжок, то це може означати, що перехід до наступного стану дії в сусідній доріжці асоційований з готовністю деякого документа (об'єкт у деякому стані).

Необхідність синхронізації виникає щораз, коли паралельно виконувані дії впливають на один на одного. На діаграмі діяльності, на відміну від діаграми станів, де використовується синхронізуючий стан, немає необхідності у додаткових позначеннях. Паралельні процеси синхронізуються переходами " поділ -злиття".

На рис 17. наведений приклад діаграми діяльності для варіанта з лабораторної роботи 1.



**Рис.17.** Приклад діаграми діяльності

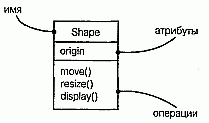
**Лабораторна робота 4.**

**Діаграма класів (class diagram).**

Моделювання системи означає ідентифікацію сутностей, важливих з тієї або іншої точки зору. Ці сутності становлять словник модельованої системи.

У мові **UML** усі сутності подібного роду моделюються як класи. Клас - це абстракція сутностей, що є частиною вашого словника.

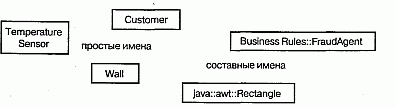
На рис. 18 наведено приклад зображення класу в мові **UML**. Таке позначення дозволяє візуалізувати абстракцію незалежно від конкретної мови програмування й підкреслити її найбільш важливі характеристики: ім'я, атрибути й операції.



**Рис. 18** Класи

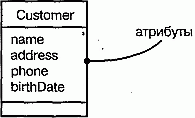
*Класом* (Class) називається опис сукупності об'єктів із загальними атрибутами, операціями, відносинами й семантикою. Графічно клас зображується у вигляді прямокутника.

У кожного класу повинне бути ім'я, що відрізняє його від інших класів. Ім'я класу - це текстовий рядок. Якщо воно записане без імені пакета воно називається простим іменем; до складеного імені попереду додане ім'я пакета, куди входить клас. Ім'я класу в межах пакету повинне бути унікальним. При графічному зображенні класу показується тільки його ім'я, як на рис. 19.

РисПростыеисоставныеимена

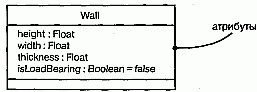
**Рис.19** Прості й складені імена

*Атрибут* - це іменована властивість класу, що включає опис множини значень, які можуть набувати екземпляри цієї властивості. У класа може бути будь-яка кількість атрибутів. Атрибут представляє деяку властивість модельованої сутності, загальну для всіх об'єктів даного класу. Таким чином, атрибут є абстракцією даних об'єкта або його стану. У кожний момент часу будь-який атрибут об'єкта, що належить даному класу, має цілком певне значення. Атрибути представлені в розділі, який розташований під іменем класу; при цьому вказуються тільки їх імена (див. рис. 20).



**Рис. 20** Атрибути

При описі атрибута можна явно вказувати його тип і початкове значення, прийняте за замовчуванням, як продемонстровано на рис.21.



**Рис. 21** Атрибути і їх типи

Для запису атрибутів класу у мові UML існує визначений формат. Спочатку записується квантор видимості, потім ім'я атрибута, його кратність, тип значень атрибута і можливо, його вихідне значення:

<квантор видимості><ім'я атрибута>[кратність]:

<тип атрибута> = <вихідне значення>{ рядок-властивість}

Для позначення квантору видимості використовуються наступні знаки:

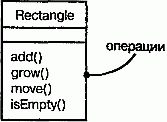
* Символ "+" - атрибут має загальнодоступний тип видимості (public).
* Символ "#" - атрибут має захищений тип видимості (protected).
* Символ "-" - атрибут має закритий тип видимості (private).

Якщо квантор видимості опущений, то це не означає, що має місце тип видимості за умовчуванням пакетний, public або private, як у деяких мовах програмування, просто інформація про тип видимості відсутня.

Замість вищевказаних символів можна також використовувати і ключові слова : public, protected, private.

*Операцією* називається реалізація послуги, яку можна запросити в будь-якого об'єкта класу для впливу на поведінку. У всіх об'єктів класу є загальний набір операцій. Кількість операцій в класі необмежена. Деякі операції можуть змінювати стан або дані об'єкта. Операції класу зображуються в розділі, розташованому нижче роздягнула з атрибутами. При цьому можна обмежитися тільки іменами, як показано на рис.22.

Більш детальна специфікація виконання операції здійснюється за допомогою приміток і діаграм діяльності.



**Рис. 22** Операції

Для запису операцій у мові UML використовується наступний синтаксис:

<квантор видимості><ім'я операції>(список параметрів):

<вираз типу значення, що повертається, >{ рядок-властивість}

Для позначення квантору видимості, як і у випадку атрибутів класу, використовуються наступні знаки:

* Символ "+" - атрибут має загальнодоступний тип видимості (public).
* Символ "#" - атрибут має захищений тип видимості (protected).
* Символ "-" - атрибут має закритий тип видимості (private).

Якщо квантор видимості опущений, то це не означає, що має місце тип видимості за умовчуванням пакетний, public або private, як у деяких мовах програмування, просто інформація про тип видимості відсутня. Також можна записувати відповідне ключове слово: public, protected, private.

Якщо операція не змінює стан системи, вона має властивість "{запит}" ("{query}"). Якщо такої властивості немає, то операція може змінити стан системи, хоча і необов'язково.

Якщо для операція позначається рядком-властивістю "{concurrency = ім'я}", де в якості імені можуть бути такі значення, як послідовна (sequential), паралельна (concurrent), охоронювана (guarded), то дана операція виконується паралельно.

Вищевказані значення визначаються таким чином:

* послідовна (sequential) — в системі може виконуватися тільни одна така операція.
* паралельна (concurrent) — в системі може виконуватися декілька таких операцій, при цьому паралельність повинна підтримуватися на рівні реалізації моделі.
* охоронювана (guarded) — звертання до такої операції строго впорядковуються у часі для збереження цілісності об'єктів даного класу.

Якщо даний рядок-властивість відсутній, то так як властивість паралельності не вказана, то припускають послідовне виконання.

Абстрактні операції позначаються ключовим словом "{abstract}", або курсивом.

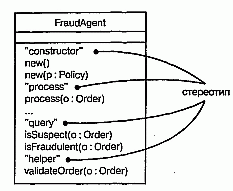
Для операції можна дати особливості її реалізації у вигляді примітки, приєднаної до запису операції у класі. Також для позначення реакції об'єктів класу на деякий сигнал у записі операції використовується ключове слово сигнал ("signal"). Таким же чином можна показати реакцію об'єктів класу на виключення (помилкові ситуації).

Примітки у мові UML являють собою прямокутник з "загнутим" верхнім правим куточком і текстом у ньому. Якщо текст є записом на деякій мові програмування, то він розміщується у дужках.

Не обов'язково вказувати тип значення, що повертається і формальні параметри. Типи операцій і атрибутів починаються з великої літери, а імена самих операцій і атрибутів - з малої. Ім'я операції і круглі дужки вказуються обов'язково.

Всі операції і атрибути класу показувати не обов'язково. Як правило, це просто неможливо - їх надто багато для одного малюнка, - та й не потрібно (оскільки для даного представлення системи лише невелика підмножина атрибутів і операцій має значення). Із цих причин клас звичайно згортають, тобто зображують лише деякі з наявних атрибутів і операцій, а то й зовсім опускають їх. Явно наявність додаткових атрибутів або операцій можна позначити, поставивши наприкінці списку три крапки.

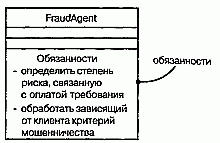
Атрибути і операції можна доповнювати стереотипами з метою більшої виразності і зрозумілості ( рис. 23).



**Рис. 23** Використання стереотипів для опису властивостей класу

*Обов'язки* (Responsibilities) класу - це певний контракт, який обов'язковий для виконання класом. При визначенні класу стверджується, що всі екземпляри класу мають однотипний стан і поведінку. Таким чином обов'язки класу виконуються саме через відповілні атрибути і операції.

Графічно обов'язки зображують в особливому розділі в нижній частині піктограми класу (див. рис. 24). Їх можна вказати також у примітці.



**Рис. 24** Обов'язки

Крім внутрішньої організації або структури класів на відповідній діаграмі вказуються різні відносини між класами. У мові UML інують такі зв'язки (відносини) між класами:

  1. Відношення залежності це таке семантичне відношення між елементами моделі, що не являє собою реалізацію, узагальнення чи асоціацію. Якщо при зміні якогось елемента моделі траби змінити інший елемент, то в такому випадку застосовають саме відношення залежності.

На діаграмі відношення залежності позначається у вигляді пунктирної лінії зі стрілкою, яка з'єднує відповідні елементи. Стрілка повинна іти від класу-клієнта до класу-джерела (незалежного класу).

Лінія може розщеплюватися на декілька ліній, якщо треба з'єднати множини елементів моделі.

Біля стрілки можна ставити індивідуальне ім'я і стереотипи (ключові слова у лапках). Для відношення залежності існують такі стереотипи:

* "access" — атрибути і операціі класу-джерела доступні для класів-клієнтів;
* "bind" — клас-клієнт може використовувати деякий шаблон для своєї наступної параметризації;
* "derive" — атрибути класу-клієнта можуть бути обчислені по атрибутах класу-джерела;
* "import" — класс-клієнт імпортує до себе всі відкриті атрибути й операції класу-джерела;
* "refine" — інформує, що клас-клієнт є уточненням класу-джерела , коли з'являється додаткова інформація в ході роботи над проектом.

2. Деяке відношення між класами можна виразити відношенням асоціації. На діаграмі відношення асоціації зображується суцільною лінією з необов'язковими елементами, такими як ім'я асоціації (з великої літери), кратність класів та їх імена.

За арністю відношення асоціації може бути бінарним, тернарним та вищої арності (N-арним). Якщо є порядок проходження класів, то він може визначатися стрілкою у формі трикутника (перший - другий).

На діаграмі N-арна асоціація зображується ромбом з вершин якого (або з середин сторін) проведені суцільні лінії до класів. Біля ромбу може стояти ім'я асоціації.

Для N-арної асоціації характерні такі особливості:

- в діаграмі не зазначається порядок класів;

- якщо від ромба іде пунктирна лінія до деякого класу, то даний клас підтримує властивості даної N-арної асоціації, а сама N-арна асоціація має атрибути, операції й/або асоціації. Таким чином така асоціація є класом, зображеним у вигляді прямокутника і називається асоціацією-класом.

Для класів, що входять в асоціацію може вказуватися ім'я ролі даного класу у вигляді тексту біля кінця асоціації для відповідного класу.

Також може бути таке позначення як кратність окремих класів, що є кінцями асоціації. Дане позначення записується як інтервал цілих чисел і означає можливу кількість окремих екземплярів або значень кортежів цієї асоціації, які можуть мати місце, коли інші N-1 екземплярів або значень класів фіксовані.

Всі інші властивості асоціації представляються як атрибути класу і записуються звичайним способом у відповідному місці прямокутника класу.

Асоціація виключення (Xor-association) розглядається як частковий випадок відношення виключення. Асоціація виключення означає, що з усіх варіантів асоціації в даний час використовується тільки один її екземпляр. Асоціація виключення на діаграмі класів позначається з рядком-обмеженням "{хог}".

Одним з видів відношення асоціації є так зване відношення агрегації. Відношення агрегації між класами встановлюється, якщо один клас включає в себе інший клас.

Фактично дане відношення використовується для позначення зв'язку типу " частина-ціле".

На діаграмі відношення агрегації позначається суцільною лінією з незафарбованим ромбом біля класу, який являє собою "ціле".

Частковим випадком агрегації є відношення композиції, яке встановлюється, якщо один клас включає в себе інший клас, при цьому дані класи не можуть існувати без існування класу, який їх включає. Зі знищенням класу, який включає в себе інші класи вони також знищуються.

Прикладом відношення композиції є вікно із розміщеною в ньому кнопкою. При закритті вікна, знищуються також і кнопка.

На діаграмі відношення агрегації позначається суцільною лінією з зафарбованим ромбом біля класу, який являє собою "ціле".

3. Відношення між більш загальним об’єктом (батьком або предком) і більш частковим або спеціальним елементом (дочірнім або нащадком) називається відношенням узагальнення. Воно має місце для класів, варіантів використання, пакетів та інших елементів мови UML.

На діаграмі класів відношення узагальнення зображується у вигляді суцільної лінії із трикутною стрілкою, направленою на більш загальний клас.

Біля стрілки може розміщуватися текст, який знаходиться у фігурних дужках і розглядається як обмеження та стосується всіх підкласів даного відношення.

Приклади ключових слів, які застосовуються як обмеження:

* {complete} — говорить, що крім уже вказаних у даному відношенні узагальнення не буде нових класів-нащадків;
* {disjoint} — підкласи не можуть містити об'єктів, які є екземплярами більше ніж одного класу. Наприклад, якщо суперклас - клієнт банку, а підкласи - фізична особа і компанія, то в даному випадку ніяка конкретна фізична особа не може бути одночасно й конкретною компанією.
* {incomplete} — говорить, що крім уже вказаних у даному відношенні узагальнення є і інші класи-нащадки;
* {overlapping} — підкласи можуть містити об'єкти, які є екземплярами більше ніж одного класу. Наприклад, якщо у класу "Багатокутник" є підкласи "Прямокутник" і "Ромб", то існує клас "Квадрат", який є одночасно і "Прямокутником" і "Ромбом".

Інтерфейси на діаграмі зображаються у вигляді прямокутника класу зі стереотипом "interface" і тільки із секцією операцій.

Для того, щоб продемонструвати зв'язки не між класами, а між об'єктами використовується так звана діаграма об'єктів. Об'єкти зображуються аналогічно класам, але ім'я об'єкту записується у формі: "ім'я об'єкта: ім'я класу" і завжди підкреслюється. Якщо ім'я об'єкту відсутнє, а є тільки ім'я класу і двокрапка, то мається на увазі анонімний об'єкт. Може також бути відсутнім і ім'я класу, а вказується тільки ім'я об'єкту. Атрибути об'єктів мають конкретні значення.

Зв'язки між об'єктами показуються суцільними лініями.

Шаблон (template) або параметризований клас (parametrized class) являє собою клас, який в якості параметрів приймає типи, наприклад, цілі числа, дійсні числа, рядки, масиви, класи та ін. Таким чином операції, які визначені в шаблоні, будуть однаково обробляти будь-які типи даних, що передані до шаблону (узагальнене програмування).

Шаблон на діаграмі позначається аналогічно звичайному класу але до верхнього правого кута прямокутника додається невеликий прямокутник із пунктирних ліній, в якому розміщується список формальних параметрів типів.

4. Відношення реалізації. Інтерфейс - це набір операцій, які описують послуги, надавані класом або компонентом. Істотним є відношення між компонентом і інтерфейсом. Усі популярні компонентні засоби операційних систем (такі як **СОМ+**, **CORBA** і **Enterprise Javabeans**) використовують інтерфейси для "склеювання" різних класів.

Користуючись кожним з таких засобів, можна побудувати декомпозицію фізичної реалізації шляхом специфікування інтерфейсів, що представляють основні стикувальні вузли системи. Потім ви можете надати класи, що реалізують інтерфейси, поряд з іншими класами, які через ці інтерфейси одержують доступ до надаваних послуг.

Відношення між класом і його інтерфейсами можна представити за допомогою двох способів (рис 25). У першому випадку інтерфейс представляється в згорнутій (**elided**) формі. Між класом, що реалізує інтерфейс і самим інтерфейсом має місце відношення згорнутої реалізації. У другому випадку інтерфейс зображується в розгорнутому вигляді, інколи з розкриттям операцій. Клас, що реалізує інтерфейс, приєднується за допомогою відношення повної реалізації. В обох випадках клас, що одержує доступ до послуг інших класів через цей інтерфейс, підключається до нього за допомогою відношення залежності.

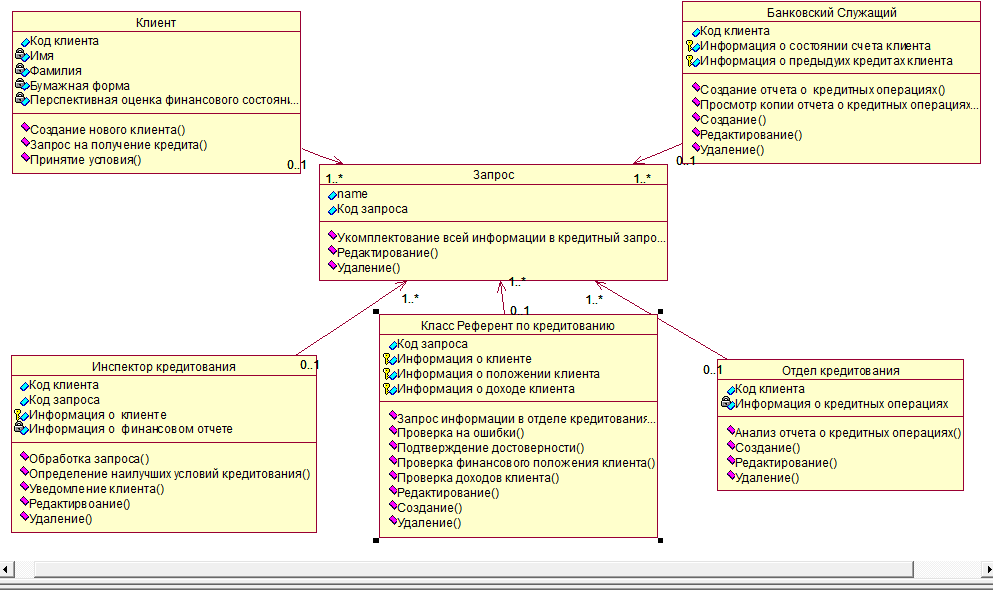


**Рис.25** Класи й інтерфейси

Інтерфейс, реалізований класом, називається *експортованим інтерфейсом* (**Export interface**). Це означає, що клас через даний інтерфейс надає ряд послуг іншим класам. Інтерфейс, яким клас користується, називається *імпортованим* (**Import interface**). Це означає, що клас сумісний з таким інтерфейсом і залежить від нього при виконанні своїх функцій.

Інтерфейси перетинають межі між логічними й фізичними сутностями. Той же самий інтерфейс, який використовується або реалізується класом, буде використовуватися або реалізовуватися класами, які реалізує цей клас.

На рис 26. наведений приклад діаграми класів для варіанта з лабораторної роботи 1.



**Рис.26.** Приклад діаграми класів.

**Лабораторна робота 5.**

**Діаграма компонетів (component diagram).**

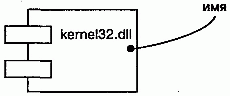
В **UML** усі фізичні сутності моделюються у вигляді компонентів. Компонент (**Component**) - це фізична сутність, що реалізує деякий набір інтерфейсів. Таким чином, інтерфейси утворюють міст між логічними й фізичними моделями.

Компоненти можуть використовуватися не тільки для моделювання сутностей, але й для представлення інших елементів працюючої системи - приміром, таблиць, файлів і документів.

Діаграма компонентів створюється для вирішення таких завдань:

* Представлення у графічному вигляді загальної структури вихідного коду програмної системи.
* Специфікації реалізованого варіанта програмної системи.
* Можливість використання окремих фрагментів програмного коду багато разів .
* Створення фізичної й концептуальної схем баз даних.

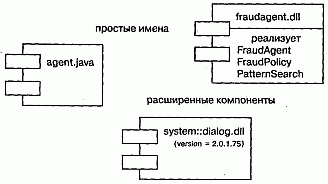
Компонент в **UML**  треба представляти так, як зображено на рис. 27. Таким чином незалежно від операційної системи або мови програмування можна представини компонент. Використати цю нотацію для представлення конкретних видів компонентів можна за допомогою одного з механізмів розширення **UML -** стереотипів .

РисКомпоненты

**Рис. 27** Компоненти

*Компонент* (Component) - це фізична замінна частина системи, сумісна з одним набором інтерфейсів, що й забезпечує реалізацію якого-небудь іншого. Компонент зображується у вигляді прямокутника із вкладками.

Кожен компонент повинен мати ім'я, відмінне від імен інших компонентів. Ім'я - це текстовий рядок, який ідентифікує даний компонент. Якщо у склад імені не входить ім'я пакету,в якому перебуває компонент, воно називається простим, якщо входить - складеним. Ім'я компонента повинне бути унікальним усередині охоплюючого пакета. Звичайно при зображенні компонента вказують тільки його ім'я, як видно з рис. 28. Можна також для показу деталей використовувати позначені значення або додаткові розділи.



**Рис. 28** Просте й розширене зображення компонентів

Існують наступні види компонентів:

1. *Компоненти розгортання* (**Deployment components**), які безпосередньо використовуються у побудові системи. До них належать бібліотеки, що динамічно **підключаються** (DLL) і готові до виконання програми (EXE). Визначення компонентів в **UML** охоплює такі традиційні об'єктні моделі як **Enterprise Javabeans, CORBA,** **СОМ+** і інші, що можливо містять таблиці бази даних , динамічні Web-сторінки, і готові до виконання модулі, в яких застосовуються закриті способи обміну інформацією.

2. *Компоненти - робочі продукти* (**Work product components**). Вони з'являються у результаті процесу розробки. Наприклад, це файли з вихідними текстами програм і даними, які використовуються при створенні компонентів розгортання. Ці компоненти безпосередньо не використовуються у функціонуванні системи, що виконується, але є робочими продуктами, з яких система створюється.

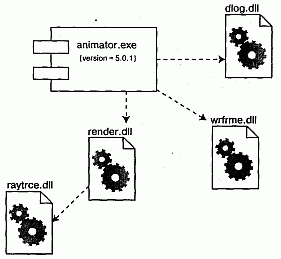
3. *Компоненти виконання* (**Execution components**). Такі компоненти виникають при роботі системи. Наприклад, це екземпляр об'єкту **СОМ+**, який створюється з **DLL**.

До компонентів застосовні усі механізми розширення **UML**. Часто для розширення властивостей компонентів застосовуються помічені значення (для показу версії компонента) і для завдання нових видів компонентів використовуються.

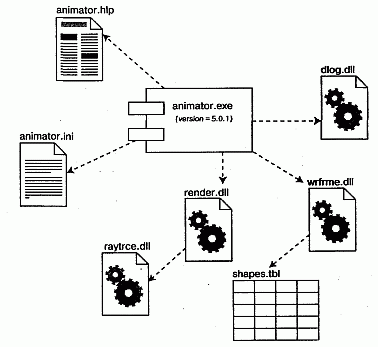
До компонентів в **UML** застосовуються п'ять стандартних стереотипів:

* file (файл) - описує компонент, що є документом з вихідним кодом або даними;
* document (документ) - описує компонент, що представляє документ.
* executable (здійсненний) - описує компонент, який може виконуватися у вузлі;
* library (бібліотека) - бібліотека підпрограм (статична або динамічна);
* table (таблиця) - описує компонент, що представляє таблицю бази даних;

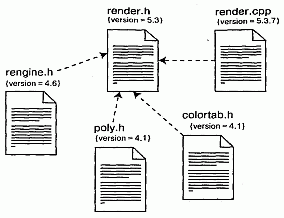
Для зображення компонентів використані стандартні елементи для програм, що виконуються, і бібліотек. На діаграмі представлені також залежності між компонентами (мал.29- 31).



**Рис. 29** Моделювання програм, що виконуються, і бібліотек.

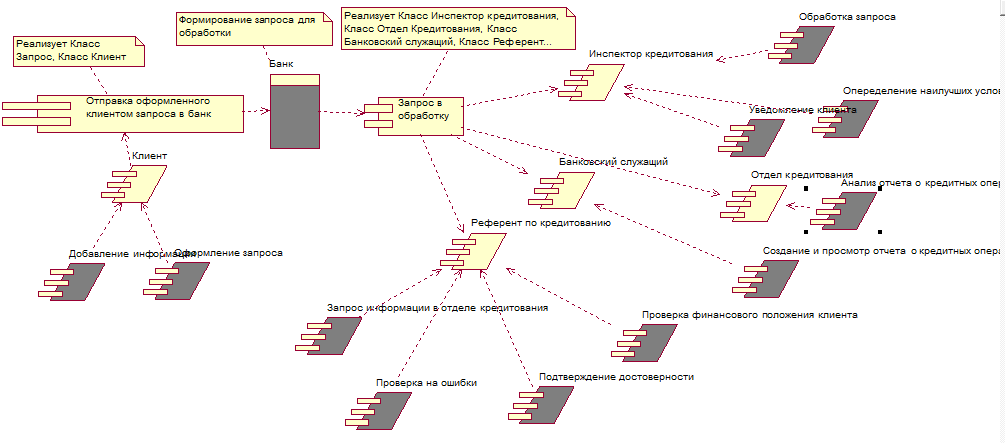


**Рис.30.** Моделювання таблиць, файлів і документів



**Рис.31** Моделювання вихідного коду

На рис 32. наведений приклад діаграми компонентів для варіанта з лабораторної роботи 1.



**Рис.32.** Приклад діаграми компонентів.

**Лабораторна робота 6.**

**Діаграма розгортання (deployment diagram).**

На діаграмі розгортання показуються тільки ті елементи, які мають місце під час виконання програми, наприклад, виконувані файли або динамічні бібліотеки. Ішні компоненти, такі як вихідні тексти програм зображуються на інших типах діаграм, таких як діаграми компонентів.

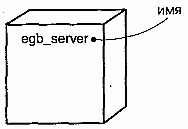
Взагалі, на діаграмі розгортання забражуються процеси, пристрої та процесори із взаємозв'язками. Діаграма розгортання повинна вміщувати всю систему вцілому і є завершує процес об'єктно-орієнтованого аналізу і проектування.

Основні призначення діаграми розгортання можна виразити трьома пунктами:

* Визначити які компоненти знаходяться в яких фізичних вузлах .
* Між вузлами реалізації зобразити зв'язки, які мають місце під час роботи системи.
* Знайти недоліки системи і змінити її таким чином, щоб підвищити продуктивність.

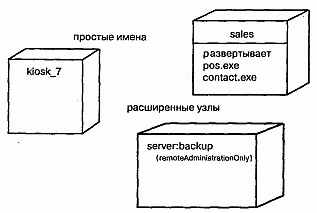
Вузлом називається деякий елемент системи, який має обчислювальний або інший ресурс, наприклад, процесор, принтер, датчик, сканер, модем або інший електронний пристрій.

Графічне зображення вузла в **UML** показане на рис. 33. Це канонічне позначення дозволяє візуалізувати вузол, не конкретизуючи його апаратуру. За допомогою стереотипів - одного з механізмів розширення **UML** - можна адаптувати цю нотацію для представлення конкретних процесорів і пристроїв.

РисУзлы

**Рис.33** Вузли

Вузол має унікальне просте або складене ім'я (тобто ім'я пакета+просте ім'я) (рис. 34). Разом з іменами вузлів для більшої деталізації можна використовувати позначені значення чи додаткові розділи.



**Рис. 34** Просте й розширене зображення вузлів

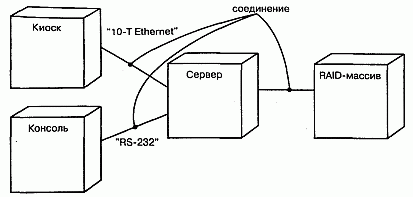
Явно зобразити компоненти у вузлі можна записавши їх імена нижче горизонтальної лінії. Також можна забраження компонентів безпосередньо вкласти у вузол.

На діаграмі розгортання показуються також відносини між вузлами. Це можуть бути як фізичні з'єднання, так і залежності.

Крім зображень вузлів на діаграмі розгортання вказуються відносини між ними. У якості відносин виступають фізичні з'єднання між вузлами й залежності між вузлами й компонентами, зображення яких теж можуть бути присутнім на діаграмах розгортання.

На діаграмі з'єднання показуються у вигляді ліній без стрілок і являють собою вид асоціації. Як правило, з'єднання означає наявність фізичного каналу по якому здійснюється передача інформації між вузлами. Також до з'єднань широко використовуються примітки (рис

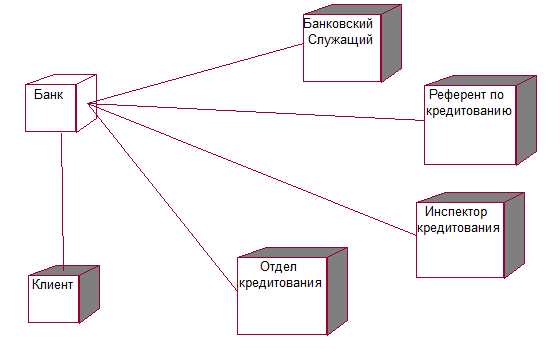
. 35).



**Рис.35** З'єднання

Якщо кількість компонентів у вузлі занадто велике, можна винести їх зображення поза вузол і з'єднати їх з вузлом відношенням залежності.

На рис 36. наведений приклад діаграми компонентів для варіанта з лабораторної роботи 1.



**Рис.36.** Приклад діаграми розгортання.

**ЗАВДАННЯ**

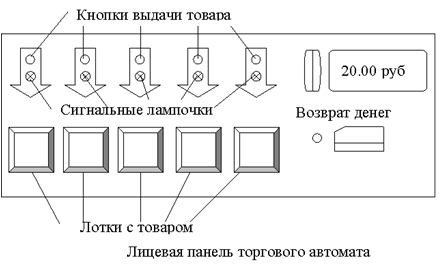
У кожному із запропонованих варіантів потрібно за допомогою Case-засобу Rational Rose побудувати модель програмного забезпечення**.**

**Варіант 1. Магазин відеопрокату**

Відеомагазин потребує комп'ютерної системи. Асортименти магазинустановитьблизько тисячі відеокасет і п'ятсот відео-дисків. Запас уже замовлений в одного постачальника, однак, для майбутніх замовлень директор має намірзалучити до послуг більше число постачальників. Усі відеокасети й диски постачені штрих-кодом, так що сканер, інтегрований у систему, може підтримувати операції видачі напрокат і повернення відеофільмів. Членські картки клієнтів також постачені штрих-кодом.  
Клієнти мають можливість резервувати відео таким чином, щоб комплект відеофільмів був зібраний до певної дати. Система повинна мати пошуковий механізм для відповідей на запити клієнтів, включаючи питання, що стосуються фільмів, яких немає в асортиментах магазину (але які він може замовити на прохання клієнта). Для кожного фільму встановлений конкретний період прокату (обчислювальний у днях) з відповідною платою за прокат за цей період. Відеомагазин повинен бути у змозі негайно дати відповідь на будь-який запит з наявності фільмів у запасі, а також кількості касет або дисків (поточні умови по кожній стрічці й диску повинні бути відомі й зафіксовані). Плата за прокат відрізняється залежно від відеоносія: касета або диск. Хоча магазин тримає в запасі відеодиски тільки одного формату – DVD, користувачі бажали б розширити в майбутньому систему прокату й на інші формати дисків. Працівники відеомагазину прагнуть запам'ятати коди найбільш популярних стрічок. Найчастіше при ідентифікації фільму вони використовують саме код фільму, а не його назву (оскільки фільм з одним назвою міг випускатися різними режисерами). Додаткові вимоги: За касети й диски, повернуті пізніше строку, стягується додаткова плата за період, що перевищує строк прокату. Кожнийвідеоносій має унікальний ідентифікаційний номер. Фільми замовляються в постачальника, який, у загальномувипадку, може поставити касети й диски протягом одному тижня. Звичайно одне замовлення робиться на кілька фільмів. Забронювати можна ті фільми, які замовлені в постачальника й/або всі копії яких перебувають у прокаті. Можна також забронювати ті фільми, яких немає в запасі і які не замовлені в постачальника; при цьому із клієнта потрібен задаток за один період прокату. Клієнт може також зробити кілька попередніх замовлень, однак для кожного заброньованого фільму готується окремий запит на бронювання. Бронювання може бути скасоване. Через відсутність реакції з боку клієнта, більш точно, протягом одному тижня змоменту, коли клієнтові було повідомлено про можливість взяти фільм напрокат. Якщо за фільм був сплачений задаток, він записується на рахунок клієнта. База даних зберігає звичайну інформацію про постачальників і клієнтів, тобто адреси, телефонні номери і т.д. У кожномузамовленніпостачальникові вказуються фільми, що замовляються, їхкількість, форматикасети/диска, а також дата очікуваноїдоставки, відпускнаціна, можливі знижки іт.д.  Коли касета повертається клієнтом або надходить від постачальника, спочатку задовольняються попередні замовлення. Працівники магазину встановлюють контакт із клієнтами, що зробили попереднє замовлення. Для правильної обробки бронювання фільмів інформація, пов'язана із бронюванням, обновлюється двічі: після встановлення контакту із клієнтом, коли йому повідомляється, що «заброньований фільм прийшов», і після здачі фільму клієнтові напрокат. Ці кроки гарантують правильне проведення операції бронювання. Клієнт може взяти кілька касет або дисків, однак кожному взятому відеоносієві ставиться у відповідність окремий запис. Для кожного видаваного напрокат фільму фіксуються дата й час видачі, установлений і фактичний строк повернення. Пізніше запис про прокатобновлюється, щоб відбити факт повернення відеофільму й факт остаточного платежу (або повернення грошей). Крім того, запис зберігає інформацію про продавця, відповідального за прокат фільму. Детальна інформація про клієнта й по прокату зберігається протягом року, щоб можна було легко визначитирівень довіри до клієнта. Стара інформація із прокату зберігається протягом року з метою проведення аудита.  
Усі операції виконуються з використанням готівки, електронного переказу грошей або кредитних карток. Від клієнтів потрібно внести плату за прокат при видачі касет/дисків. Якщо касета/диск повернуті пізніше встановленого строку (або не можуть бути повернуті за якимись причинами), плата знімається або зрахунку клієнта, або ухвалюється безпосередньо від клієнта. Якщо касета/диск затримані більш ніж на два дні, клієнтові відправляється повідомлення про затримку. Після відправлення двох повідомлень про затримку однієї й тієї ж касети/диска, клієнт попереджується про те, що він є «порушником» і при наступному його зверненні до магазинукерівництво розглядає питання про зняття з нього статусу «порушника».

**Варіант 2. Торговельний автомат**

Потрібно розробити засобами Rational Rose модель програмного забезпечення вбудованого процесора універсального торговельного автомата.



Зовнішній вигляд автомата зображений на малюнку. В автоматі є п'ять лотків для зберігання й видачі товарів. Завантаження товарів на лотки здійснюється обслуговуючим персоналом. Автомат стежить за наявністю товару. Якщо який-небудь товар розпроданий, автомат відправляє повідомлення про це на станцію обслуговування й інформує покупців (запалюється червона лампочка поруч із лотком даного товару). Автомат приймає до сплати паперові купюри й монети. Спеціальний індикатор показує поточну суму грошей, прийнятих автоматом до оплати. Після введення грошей клієнт натискає на кнопку видачі товару. Видача товару проводиться тільки в тому випадку, якщо введена сума грошей відповідає ціні товару. Товар видається поштучно. При натисканні на кнопку «Повернення» клієнтовівертаються всі прийняті від нього до оплати гроші. Повернення грошей не проводилося після видачі товару. Автомат повинен коректно працювати при одночасномунатисканні на кнопки видачі товару й повернення грошей. У спеціальномувідділенні автомата, що закривається замком, є «секретна кнопка», яка використовується обслуговуючим персоналом для вилучення виторгу. При натисканні на цю кнопку відкривається доступ до ящика із грішми.

Автомат одержуєзі станції обслуговування дані про товари й зберігає їх у своїй пам'яті. Дані містять у собі ціну, найменування товару, номер лотка, на якомуперебуває товар і кількість товару на лотку. Варіант завдання містить у собі розробку схеми бази даних про товари.

**Варіант 3. Табло на станції метро**

Потрібно розробити засобами Rational Rose модель програмного забезпечення табло для інформаційної служби метрополітену. Табло розташовані на кожній станції метро. Вони працюють під керуванням єдиного пункту керування (ПК) інформаційної служби метро. Табло відображає поточний час (години, хвилини, секунди) і час, щопройшов з моменту відправлення останнього поїзда (хвилини, секунди). Момент прибуття й відправлення поїздавизначається за допомогою датчиків, установлюваних на шляхах. Усі табло метро синхронізовані,поточний час відлічується й установлюється із центральноїслужби часу, щоперебуваєна ПК. На табло показується кінцева станція призначення поїзда, що прибуває. Ці дані втримуються в розкладі руху поїздів, який зберігається в пам'яті табло й періодично обновляється з ПК. В « біжучому рядку » табло відображається рекламнаінформація. Пам'ять табло зберігає до 10 рекламних повідомлень. Повідомлення відображаються один за одним з невеликими паузами, циклічно. Зміст рекламних повідомлень надходить із ПК. Додаткова функція табло – по запиту з ПК воно пересилає дані про порушення у розкладі (передчасних відправленнях поїздів або запізненнях). У ході виконання завдання повинна бути створена схема бази даних для зберігання рекламних повідомлень, розкладу й відомостей про порушення розкладів. *Пояснення: у завданні потрібно розробити модель ПО тільки для табло, але не для пункту керування інформаційної служби.*

**Варіант 4. Система автоматизації для пункту прокату відеокасет**

Потрібно розробити засобами Rational Rose модель програмної системи автоматизації роботи пункту прокату відеокасет (далі в тексті – системи).

Пункт прокату містить каталог касет, що є в наявності в цей момент часу. Система підтримує роботу каталогу, дозволяючипрацівникам прокату додавати нові найменування касет, видаляти старі й редагувати дані про касети. Клієнт, що звернувся в пункт, вибирає касету по каталогу, вносить заставу й забирає її на певний строк. Строк прокату, вимірюваний у добі, визначається при видачі касети. Вартість прокату обчислюється системою виходячи з тарифу за добу й строку прокату. Клієнт повертає касету й сплачуєза прокат. Якщо касета не ушкоджена, клієнтовіповертаєтьсязастава. Службовець пункту прокату реєструє здачу касети клієнтові і її повернення в системі. Якщо клієнт ушкодив касету, то касета видаляється з каталогу, а заставазалишається в касі прокату.

При необхідності службовець може запросити в системи наступні дані:

* є чи в наявності касета з даною назвою;
* коли буде повернута яка-небудь касета з тих, що здані в прокат;
* чи є даний клієнт постійним клієнтом пункту прокату ( чи користувався прокатом 5 або більшразів).

Постійним клієнтам надаються знижки, а також від них приймаються заявки на поповнення асортиментів касет. Заявки реєструються в системі. Згідно них готується підсумковий звіт, керуючись яким працівники пункту прокату обновляють асортименти касет.

Необхідно розробити схему бази даних для зберігання каталогу, облікових записів про прокат касет і заявок на поповнення асортиментів.

**Варіант 5. Міні-АТС**

Потрібно розробити засобами Rational Rose модель програмного забезпечення вбудованого мікропроцесора міні-АТС деякої установи (автоматичної телефонної станції). Міні-АТС здійснює зв'язок між службовцями установи. Кожний абонент підключений до неї лінією зв'язку. Міні-АТС сполучає лінії абонентів (здійснює комутацію ліній). Абоненти мають номери, що складаються із трьох цифр. Спеціальний номер **«9»** зарезервований для зовнішнього зв'язку. Телефонне з'єднання абонентів проводиться в такий спосіб. Абонент піднімає трубку телефону, і міні-АТС одержує сигнал **«Трубка»**. У відповідь міні-АТС посилає сигнал **«Тон»**. Прийнявши цей сигнал, абонент набирає телефонний номер (посилає три сигнали **«Цифра»**). Міні-АТС перевіряє готовність абонента, який викликається. Якщо абонент не готовий (його лінія зайнята), міні-АТС посилає абонентові, який викликає, сигнал **«Зайнято»**. Якщо абонент готовий, міні-АТС посилає обом абонентам сигнал **«Виклик»**. При цьому телефон абонента , який викликається, починає дзвонити, а абонент, який викликає, чує в трубці довгі гудки. Абонент , який викликається, знімає трубку, і міні-АТС одержує від нього сигнал **«Трубка»**, після чого здійснює комутацію лінії. Абоненти обмінюються сигналами **«Дані»**, які міні-АТС повинна передавати від одного абонента до іншого. Коли один з абонентів опускає трубку, міні-АТС одержує сигнал **«Кінець»** і посилає іншому абонентові сигнал **«Тон»**. У будь-який момент абонент може покласти трубку, при цьому міні-АТС одержує сигнал **«Кінець»**. Після одержання цього сигналу сеанс обслуговування абонента завершується. Якщо абонент бажає з'єднатися з абонентом за межами установи, то він набирає номер **«9»**. Міні-АТС посилає по лінії, що з'єднує із зовнішньою (міською) АТС, сигнал **«Трубка»** і надалі служить посередником між телефоном абонента й зовнішньою АТС. Вона приймає й передає сигнали й дані між ними, не вносячи ніяких змін. Єдине виключеннястосується завершення сеансу. Одержавши від міської АТС сигнал **«Кінець»**, міні-АТС посилає абонентові сигнал **«Тон»** і чекає сигналу **«Кінець»** для завершення обслуговування абонента. Якщо абонент, щовикликав, першим кладетрубку, то міні-АТС одержує**сигнал «Кінець**» і передає його міський АТС і завершуєсеанс. Міні-АТС може одержати сигнал **«Виклик»** від міський АТС. Це відбувається, коли немає з'єднань із зовнішніми абонентами. Сигнал **«Виклик»** від міський АТС передається абонентові з кодом **«000»**. Тільки цей абонент може відповідати на зовнішні дзвінки.

**Варіант 6. Система складського обліку**

Система складського обліку – програмна система, що стосується всіх аспектів, пов'язаних з рухом товару на склад і зі складу. За результатами аналізу можна виділитисім основних функцій системи:

* *Облік замовлень*. Приймання замовлень від клієнтів і відповіді на запити клієнтів про стан замовлень.
* *Ведення рахунків*. Напрямок рахунків клієнтам і відстеження платежів. Приймання рахунків від постачальників і відстеження платежів постачальникам.
* *Відвантаження зі складу*. Складання специфікацій на комплектацію товарів, що відправляються зі складу клієнтам.
* *Складський облік*. Постановка товарів, що прибувають, на облік і зняття товарів з обліку при відправленні замовлень.
* *Закупівлі*. Замовлення товарів постачальникам і відстеження поставок.
* *Приймання товарів*. Прийняття на склад товарів від постачальників.
* *Планування*. Випуск звітів, у тому числі, що відбивають тенденції попиту на окремі види товарів і активність постачальників.

Стратегія з проникнення компанії, що займається торгівлею по каталогах, на нові ділянки ринку, вимагає створення ряду відносно автономних регіональних складів продукції. Кожний такий склад відповідає за облік товарів і виконання замовлень. З метою підвищення ефективності своєї роботи склад зобов'язаний сам підтримувати ту номенклатуру товарів, яка найкращим чином відповідає потребам місцевого ринку. Номенклатура може бути різною для кожного регіону. Крім того, номенклатура повинна оперативно змінюватися відповідно до мінливих потреб клієнтів. Головна компанія прагне мати на всіх складах однакові системи обліку. Основними функціями системи є:

* Облік товарів, що приходять від різних постачальників, під часїхприйняття на склад.
* Облік замовлень у міру їх надходження із центральної віддаленої організації; замовлення також можуть прийматися поштою. Їхня обробка ведеться на місцях.
* Генерація вказівокперсоналу, зокрема, про пакування товарів.
* Генерація рахунків і відстеження оплат.
* Генерація запитів про постачання й відстеження платежів постачальникам.

Крім автоматизації стандартних складських операцій, система також повинна надавати багаті можливості по генерації різних форм звітності, у тому числі, що відзеркалюють тенденції розвитку ринку, списків найбільш надійних і ненадійних постачальників і клієнтів, матеріалів для рекламних компаній.

**Варіант 7. Система торговельної компанії**

Компанія – торговельний посередник – продає товари різних виробників. Для забезпечення своєї діяльності вона потребує програмної системи обробки замовлень. Двічі на рік компанія публікує каталог продуктів, який розсилається клієнтам і іншим зацікавленим особам. Клієнти отримують товари, направляючи в компанію перелік продуктів з інформацією про оплату. Компанія виконує замовлення й відправляє товари по адресах клієнтів. Система повинна відслідковувати замовлення від моменту його одержання до відправлення товару. Клієнти можуть повертати товари, сплачуючи, можливо, при цьому деякі витрати. Деякі клієнти користуються замовленням товарів через Інтернет. Компанія користується послугами різних транспортних і страхових компаній.

**Варіант 8. Керування контактами із клієнтами**

Компанія, що займається дослідженням ринку, має стабільну клієнтську базу організацій, які купують звіти з аналізу ринку. Деякі великі клієнти купують у компанії також спеціалізоване ПЗ, призначене для створення звітів. Цих клієнтів компанія також забезпечує неопрацьованою й агрегованою інформацією для генерації їх власних звітів. Нова система керування контактами повинна перебувати в розпорядженні всіх працівників компанії, але з наданням різного рівня доступу. Співробітники відділу обслуговування клієнтів беруть шефство над системою. Система повинна забезпечити гнучке планування й перепланування видів діяльності, пов'язаних з контактами. Система підтримує функції «постійного контакту» з наявною й потенційною клієнтською базою так, щоб відгукуватися на її потреби й одержувати нові контракти на придбання товарів. Система зберігає імена, номери телефонів, звичайні поштові й кур'єрські адреси і т.д.  організацій і контактних осіб у цих організаціях. Система дозволяє співробітникам планувати завдання й заходи, які необхідно провести відносно контактних осіб. Співробітники планують завдання й заходи для інших співробітників або для себе. Завдання – це група заходів, які здійснюються для досягнення певного результату. Результатом може бути перетворення потенційного клієнта в клієнта, організація доставки товару або вирішення проблеми клієнта. До звичайних типів заходів відносяться телефонний дзвінок, візит, відправлення факсу, організація навчання і т.д.  Клієнт розглядається як наявний, якщо існує контракт із цим клієнтом на постачання товарів або послуг. Однак, функції керування контрактами виходять за межі системи. Система дозволяє виробляти різні звіти по контактах на основі поштової й кур'єрської адреси (наприклад, знаходити всіх клієнтів по поштовому коду). Дата й час створення завдання фіксуються. Можна також зберегти значення «доходу», очікуваного від здійснення завдання. Заходи для співробітника відображаються на екрані його комп'ютера у вигляді сторінки календаря (один день на сторінку). Пріоритет кожного заходу (низький, середній, високий) візуально виділяється на екрані. Не з усіма заходами зв'язане поняття «строк виконання» – деякі з них є «безстроковими» (вони можуть виконуватися в будь-який час протягом дня, на який вони заплановані). Час створення заходу не може змінюватися, а строк виконання – може. По завершенню заходу дата й час його завершення фіксуються. Система також зберігає відмітні ознаки для співробітників, які створюють завдання й заходи, яким заплановано здійснення заходу («доручене співробітникові») і які завершили захід.

**Варіант 9. Банкомат**

Потрібно розробити засобами Rational Rose модель програмного забезпечення банкомата. Банкомат – це автомат для видачі готівки по кредитних пластикових картках. До його складу входять наступні пристрої: дисплей, панель керування із кнопками, приймач кредитних карт, сховище грошей і лоток для їхньої видачі, сховище конфіскованих кредитних карт, принтер для печатки довідок. Банкомат підключений до лінії зв'язку для обміну даних з банківським комп'ютером, що зберігаєвідомості про рахунки клієнтів. Обслуговування клієнта починається з моментурозміщення пластикової картки в банкомат. Після розпізнавання типу пластикової картки, банкомат видає на дисплей запрошення ввести персональний код. Персональний код являє собою чотиризначне число. Потім банкомат перевіряє правильність уведеного коду. Якщо код зазначений невірно, користувачевінадаються ще дві спроби для введення правильного коду. У випадку повторних невдач карта переміщається в сховище карт, і сеанс обслуговування закінчується. Після введення правильного коду банкомат пропонує користувачеві вибрати операцію. Клієнт може або зняти готівкузрахунку, або довідатисяпро залишок на його рахунку. При знятті готівки зрахунку банкомат пропонує вказати суму (10, 50, 100, 200, 500, 1000 гривень). Після вибору клієнтом суми банкомат запитує, чи потрібно друкувати довідку по операції. Потім банкомат надсилає запит на зняття обраної суми центральному комп'ютеру банку. У випадку одержання дозволу на операцію, банкомат перевіряє, чи є необхідна сума в його сховище грошей. Якщо він може видати гроші, то на дисплей виводиться повідомлення «Вийміть карту». Після видалення картки із приймача, банкомат видає зазначену суму в лоток видачі. Банкомат друкує довідку щодозробленої операції, якщо вона була затребувана клієнтом. Якщо клієнт хочедовідатисяпро залишок на рахунку, то банкомат надсилає запит центральному комп'ютеру банка й виводить суму на дисплей. На вимогу клієнта друкується й видається відповідна довідка. У спеціальномувідділенні банкомата, що закривається замком, є "секретна кнопка", яка використовується обслуговуючим персоналом для завантаження грошей. При натисканні на цю кнопку відкривається доступ до сховища грошей і конфіскованим кредитним картам.

**Варіант 10. Система табельного обліку**

Завдання контролю робочого часу, виконувані відділом кадрів, включають збір видрукуваних табелів, перевірку повноти й правильності заповнення табелів, одержання при необхідності підтвердження від менеджера й введення даних табельних карток у діючу систему розрахунків заробітної плати, щоб підготувати друк платіжної відомості. Крім того, штат відділу кадрів повинен підтримувати коди зарплати (звичайний час, понаднормовий час, відпустка через хворобу і т.д.) і розподіляти список кодів серед працівників компанії при кожній його зміні.  
Ця робота у свою чергу впливає на здатність відділу кадрів виконувати інші обов'язки, у тому числі готувати звіти по зарплаті, займатисядопомогою й виконувати функції наймання, просування й звільнення. Успішневирішення проблеми повинне скоротити час, який відділ кадрів витрачає на завдання, пов'язані з табельним обліком, на 60% і більше. Зацікавленими особами в цьомупроектієпрацівники компанії, які мають наступні посадові обов'язки:

|  |  |
| --- | --- |
| Зацікавленаособа | Обов'язки |
| Адміністратор відділу кадрів | Вести облік службовців. Розглядати й вводити дані табелів службовців. Підтримувати й розподіляти коди зарплати. Готувати звіти про зарплату. Звіряти дані про зарплату. Управлятипосібниками. Консультувати службовців з питань посібників. Спостерігати за процесом наймання й відповідним обліком. Спостерігати за процесом просування й відповідним обліком. Спостерігати за процесом звільнення й відповідним обліком. Реагувати на скарги службовців. |
| Менеджер відділу кадрів | Керувати відділом кадрів. |
| Службовці | Надавативідділу кадрів інформацію, пов'язануз працівникомі їх робочимчасом. |

Користувачі системи і їх обов'язки:

|  |  |
| --- | --- |
| Користувач | Обов'язки |
| Адміністратор відділу кадрів | Розглядати й коректувати табельні картки. Підтримувати коди зарплати. |
| Менеджер відділу кадрів | Затверджувати особливі типи коректування табельних карток. |
| Службовці | Вводити дані табельних карток. |
| Менеджери | Підтверджувати табельні картки службовців. |

Нова система повинна надавати наступні функціональні можливості: Для службовця – представляти й переглядати інформацію свого табеля. Для менеджера – затверджувати табелі його службовців. Для адміністратора відділу кадрів – контролювати правильність табелів і вносити необхідні уточнення. Для адміністратора відділу кадрів – підтримувати коди зарплати.

**Варіант 11. Web-Сайт авіакомпанії**

Комерційний відділ авіакомпанії запропонував розширити свій Web-сайт, щоб дозволити користувачам:

* довідатися про виконання рейсів поточного дня;
* запросити інформацію про розклад рейсів, вартості квитків і наявності місць;
* купити квитки.

Постійні клієнти, які часто літають літаками авіакомпанії, зможуть використовувати також наступні функції:

* одержати поточну інформацію про стан свого особистого рахунку (кількість кілометрів, проведених у повітрі з початку року на дане число, кількість кілометрів, які налітав клієнт, для одержання заохочувальної винагороди (безкоштовного перельоту) і т.д.;
* купити квитки, використовуючи або інформацію про , кількість кілометрів, які налітав клієнт ( для постійних клієнтів), або кредитну картку.

Щоб гарантувати таємницю приватної інформації й запобігти несанкціонованому використанню даних про постійних клієнтів, при доступі до особистих рахунків необхідно зажадати, щоб користувач зареєструвався, увівши номер рахунку й особистий ідентифікаційний номер власника картки (PIN). Після реєстрації користувач повинен побачити початкову сторінку з урахуванням його переваг і звичок, узятих з бази даних, що зберігає інформацію про перельоти постійних клієнтів. Постійні клієнти можуть оперативно обновлювати відомості про себе. Щоб заощадити гроші, керівництво компанії вирішило використовувати низку існуючих систем:

* систему керування рахунками, що зберігає інформацію про постійних клієнтів і баланс «преміальних кілометрів»;
* маркетингову базу даних, яка відслідковує дані про виконані рейси, клас оплати й ін. (Ці дані використовуються для формування спеціальних повідомлень, які включаються в щомісячні виписки з особового рахунку постійних клієнтів.);
* базу даних тарифів;
* базу даних наявності квитків.

**Варіант 12. Система обліку товарів**

Потрібно розробити засобами Rational Rose модель системи підтримки замовлення й обліку товарів у бакалійній крамниці. У бакалійній крамниці для кожного товару фіксується місце зберігання (певна полиця), кількість товару і його постачальник. Система підтримки замовлення й обліку товарів повинна забезпечувати додавання інформації про новий товар, зміна або видалення інформації про наявний товар, зберігання (додавання, зміна й видалення) інформації про постачальників фірми, що включає в себе назву, йогоадресу й телефон. За допомогою системи складаються замовлення постачальникам. Кожне замовлення може містити кілька позицій, у кожній позиції вказуються найменування товару і його кількість у замовленні. Система обліку на вимогу користувача формує й видає на друк наступну довідкову інформацію:

* список усіх товарів;
* список товарів, що є в наявності;
* список товарів, кількість яких необхідно поповнити;
* список товарів, що поставляються даним постачальником.

У ході виконання цього варіанта завдання повинна бути розроблена схема бази даних, що зберігає інформацію про товари, замовлення й постачальниках.

**Варіант 13. Бібліотечна система**

Потрібно розробити засобами Rational Rose модель системи, що автоматизує діяльність бібліотеки. Система підтримки керування бібліотекою повинна забезпечувати операції (додавання, видалення й зміни) над даними про читачів. У реєстраційномусписку читачів зберігаються наступні відомості: прізвище, ім'я та по батькові читача; номер його читацького квитка й дата видачі квитка. Поряд з реєстраційним списком системою повинен підтримуватися каталог бібліотеки, де зберігається інформація про книги: назва, список авторів, бібліотечний шифр, рік і місце видання, назва видавництва, загальна кількість екземплярів книги в бібліотеці й кількість екземплярів, доступних у теперішній момент. Система забезпечує додавання, видалення й зміну даних каталогу, а також пошук книг у каталозі на підставі введеного шифру або назви книги. У системі здійснюється реєстрація взятих і повернутих читачем книг. Про кожну видану книгу зберігається запис про те, кому й коли була видана книга, і коли вона буде повернута. При поверненні книги в записі робиться відповідна позначка, а сам запис не видаляється із системи. Система повинна видавати наступну довідкову інформацію:

* які книги були видані за даний проміжок часу;
* які книги були повернуті за даний проміжок часу;
* які книги перебуваютьу даного читача;
* є чи в наявності деяка книга.

Варіант завдання передбачає розробку схеми бази даних, що зберігає список читачів, каталог книг і записи про видачу книг.

\* \* \*

**Варіант 14. Інтернет-Магазин**

Потрібно розробити засобами Rational Rose модель програмного забезпечення Інтернет-магазину. Виробник комп'ютерів пропонує можливість придбання своєї продукції через Internet. Клієнт може вибрати комп'ютер на Web-сторінці виробника. Комп'ютери підрозділяються на сервери, настільні й портативні. Замовник може вибрати стандартну конфігурацію або побудувати необхідну конфігурацію в діалоговомурежимі. Компоненти конфігурації (такі, як оперативна пам'ять) представляються як список для вибору з доступних альтернатив. Для кожної нової конфігурації система може підрахувати ціну.  
Щоб оформити замовлення, клієнт повинен заповнити інформацію з доставки й оплаті. У якості платіжних засобів допускається використання кредитних карток або чеків. Після введення замовлення система відправляє клієнтові по електронній пошті повідомлення з підтвердженням одержання замовлення разом із деталями, що стосуються замовлення. Поки клієнт очікує прибуття комп'ютера, він може перевірити стан замовлення в будь-який час у діалоговомурежимі. Серверна частина обробки замовлення складається із завдань, необхідних для перевірки кредитоспроможності й способу розрахунків клієнта за покупку, витребування замовленої конфігурації зі складу, печатки рахунку й подачі заявки на склад про доставку комп'ютера клієнтові. Додаткові вимоги: для знайомства зі стандартною конфігурацією обраного сервера, настільного або портативного комп'ютера клієнт використовує Web-сторінку Internet-магазину. При цьому також приводиться ціна конфігурації. Клієнт вибирає деталі конфігурації, з якими він прагне познайомитися, можливо, з наміром купити готову або скласти більшпідходящу конфігурацію. Ціна для кожної конфігурації може бути підрахована на вимогу користувача. Клієнт може вибрати варіант замовлення комп'ютера по Internet або попросити, щоб продавець зв'язався з ним для пояснення деталей замовлення, домовився про ціну й т.п.  перш, ніж замовлення буде фактично розміщене. Для розміщення замовлення клієнт повинен заповнити електронну форму з адресами для доставки товару й відправлення рахунок-фактури, а також деталями, що стосуються оплати (кредитна картка або чек). Після введення замовлення клієнта в систему продавець відправляє на склад електронну вимогу, що містить деталі замовленої конфігурації. Деталі угоди, включаючи номер замовлення, номер рахунку клієнта, відправляються по електронній пошті клієнтові, так що замовник може перевірити стан замовлення через Internet. Склад одержує рахунок-фактуру від продавця й відвантажує комп'ютер клієнтові. При виконанні цього варіанта завдання рекомендуємо ознайомитися з роботою [Коналлен-2001]. Слід визначитися, по якому архітектурному шаблонові буде будуватися Web-додаток («тонкий клієнт» або «товстий клієнт»). Відповідно до обраного шаблону слід побудувати моделі клієнтської частини магазину й серверної частини, промоделювати зв'язки між частинами додатка. Для Web-додатків типовими є наступні *класи:*

* клієнтська Web-сторінка;
* серверна Web-сторінка (наприклад, Cgi-скрипт);
* Html-Форма;
* об'єкт Javascript.

Додаткові *зв'язки* між класами Web-додатків:

* link – посилання з однієї сторінки на іншу;
* build – зв'язок між Cgi-скриптом і клієнтською сторінкою, яка генерується при його виконанні;
* submit – зв'язок між формою й серверною Web-сторінкою, що приймає дані з форми.

Типові *компоненти:*

* Web-сторінка ( Html-файл),
* Active Server Page (ASP),
* Java Server Page (JSP),
* сервлет,
* бібліотека скриптів (наприклад файл, що підключається, з Javascript-функціями).

**Варіант 15. Www-конференція**

Потрібно розробити засобами Rational Rose модель програмного забезпечення Www-Конференції. Www-Конференція являє собою сховище повідомлень у мережі Інтернет, доступ до якого здійснюється за допомогою браузера. Для кожного повідомлення конференції зберігаються значення наступних полів: номер повідомлення, автор, тема, текст повідомлення, дата додавання повідомлення, посилання на батьківське повідомлення. Початковою сторінкою конференції є ієрархічний список повідомлень. Верхній рівень ієрархії становлять повідомлення, що відкривають нові теми, а підрівністановлять повідомлення, отримані у відповідь на повідомлення верхнього рівня. Повідомлення-Відповідь завжди має посилання на вихідне повідомлення. У списку відображаються тільки теми повідомлень, їх автори й дати додавання. Переглядаючи список, користувач вибирає повідомлення й по гіперпосиланню відкриває сторінку з текстом повідомлення. Крім тексту на цій сторінці відображається список (ієрархічний) повідомлень, що є відповідями, відповідями на відповіді і т.д. Для зручності користувачів необхідно передбачити пошук повідомлень по автору або по ключових словах у темі або тексті повідомлення. Повідомлення додаються в конференцію зареєстрованими користувачами, які при відправленні повідомлення повинні вказати своє ім'я й пароль. Реєструє нових користувачів модератор конференції - її ведучий. При реєстрації користувач заповнює спеціальну форму, уміст якої потім пересилається модератору й запам'ятовується в базі користувачів. Модератор вирішує, реєструвати користувача чи ні, і відправляє свою відповідь. При додаванні повідомлень користувач має можливість почати нову тему або відповісти на раніше додані повідомлення. Після додавання повідомлення воно доступне для читання всім користувачам (навіть незареєстрованним), і список повідомлень обновлюється. Модератор має право з тих або інших причинвидаляти повідомлення будь-яких авторів. Він також може карати користувачів, що порушують правила поведінки в конференції, позбавляючи на якийсь час користувача можливості додавати й редагувати повідомлення. Варіант завдання містить у собі розробку схеми бази даних для зберігання повідомлень конференції й інформації про її учасників. Виконуючим це завдання корисно ознайомитися із заключним зауваженням до варіанта. Найбільш підходящою архітектурою для Www-конференції є «тонкий клієнт», оскільки клієнтська частина практично не містить « бізнес-логіки». Єдиним її елементом, який може виконуватися на стороні клієнта, є перевірка правильного заповнення полів форми, перед відправленням її вмісту на сервер.