Міністерство освіти і науки України

Запорізький національний університет

А. О. Лісняк, О. М. Мильцев, В. В. Мухін, О. В. Чопорова

**АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ**

**Методичні рекомендації до лабораторних занять   
для здобувачів ступеня вищої освіти бакалавра   
спеціальності 121 «Інженерія програмного забезпечення»   
освітньо-професійної програми «Програмна інженерія»**

Затверджено

вченою радою ЗНУ

Протокол № 13

від 28.06.2022

Запоріжжя

2022

УДК: 519.8(076)

ББК

Лісняк А. О., Мильцев О. М., Мухін В. В., Чопорова О. В. Архітектура та проектування програмного забезпечення : методичні рекомендації до лабораторних занять для здобувачів ступеня вищої освіти бакалавра спеціальності 121 «Інженерія програмного забезпечення» освітньо-професійної програми «Програмна інженерія». Запоріжжя : ЗНУ, 2022. 53 с.

Методичні рекомендації містять теоретичний матеріал з кожної теми дисципліни «Архітектура та проектування програмного забезпечення», хід виконання та приклади виконання лабораторних робіт, які пропонуються при вивченні даного курсу. Кожна лабораторна робота супроводжується необхідними прикладами та поясненнями до них, певними практичними завданнями, виконання яких поглиблює сприйняття матеріалу та контрольними запитаннями.

Методичні рекомендації з дисципліни «Архітектура та проектування програмного забезпечення» призначені для студентів спеціальності 121 «Інженерія програмного забезпечення» освітньо-професійної програми «Програмна інженерія».

Рецензент *С. М. Гребенюк, д. т. н., доцент*

завідувач кафедри фундаментальної та прикладної математики

Відповідальний за випуск *А.О. Лісняк, к. ф-м. н., доцент*

завідувач кафедри програмної інженерії

**ЗМІСТ**

[ВСТУП 4](#_heading=h.30j0zll)

[Загальні рекомендації до виконання лабораторних робіт 5](#_heading=h.1fob9te)

[Лабораторна робота №1. Складні системи та основні принципи об'єктно-орієнтованого підходу 6](#_heading=h.3znysh7)

[Завдання до лабораторної роботи №1 8](#_heading=h.2et92p0)

[Запитання для самоперевірки 8](#_heading=h.tyjcwt)

[Лабораторна робота №2. Формулювання призначення і цілей створення інформаційної системи 10](#_heading=h.3dy6vkm)

[Завдання до лабораторної роботи №2 11](#_heading=h.1t3h5sf)

[Запитання для самоперевірки 11](#_heading=h.4d34og8)

[Лабораторна робота №3. Визначення переліку акторів та прецедентів 12](#_heading=h.2s8eyo1)

[Завдання до лабораторної роботи №3 15](#_heading=h.17dp8vu)

[Запитання для самоперевірки 15](#_heading=h.3rdcrjn)

[Лабораторна робота №4. Опис прецедентів 17](#_heading=h.26in1rg)

[Завдання до лабораторної роботи №4 18](#_heading=h.lnxbz9)

[Запитання для самоперевірки 19](#_heading=h.35nkun2)

[Лабораторна робота №5. Створення діаграми прецедентів 20](#_heading=h.1ksv4uv)

[Завдання до лабораторної роботи №5 25](#_heading=h.44sinio)

[Запитання для самоперевірки 26](#_heading=h.2jxsxqh)

[Лабораторна робота №6. Створення діаграми послідовності 27](#_heading=h.z337ya)

[Завдання до лабораторної роботи №6 34](#_heading=h.3j2qqm3)

[Запитання для самоперевірки 34](#_heading=h.4i7ojhp)

[Лабораторна робота №7. Створення діаграми класів 35](#_heading=h.2xcytpi)

[Завдання до лабораторної роботи №7 40](#_heading=h.3whwml4)

[Запитання для самоперевірки 40](#_heading=h.2bn6wsx)

[Лабораторна робота №8. Створення діаграми діяльності 41](#_heading=h.qsh70q)

[Завдання до лабораторної роботи №8 43](#_heading=h.1pxezwc)

[Запитання для самоперевірки 43](#_heading=h.49x2ik5)

[Лабораторна робота №9. Створення діаграми компонентів 45](#_heading=h.2p2csry)

[Завдання до лабораторної роботи №9 46](#_heading=h.3o7alnk)

[Запитання для самоперевірки 47](#_heading=h.23ckvvd)

[Лабораторна робота №10. Створення діаграми розгортання 48](#_heading=h.ihv636)

[Завдання до лабораторної роботи №10 50](#_heading=h.32hioqz)

[Запитання для самоперевірки 50](#_heading=h.1hmsyys)

[РЕКОМЕНДОВАНА ЛІТЕРАТУРА 52](#_heading=h.41mghml)

# 

# ВСТУП

Впровадження комп‘ютерів практично у всі сфери життя потребує від сучасного спеціаліста в галузі комп‘ютерних технологій вміння розробляти архітектуру програмного забезпечення для широкого класу застосувань, забезпечуючи ефективне використання комп‘ютерних систем.

Метою вивчення навчальної дисципліни «Архітектура та проєктування програмного забезпечення» є отримання студентами теоретичних знань та набуття умінь з основних принципів проектування програмного забезпечення, шаблонів проектування, середовищ розробки і архітектур програмного забезпечення.

Основними завданнями вивчення дисципліни «Архітектура та проєктування програмного забезпечення» є оволодіння різними архітектурами програмного забезпечення, основними принципами проєктування програмного забезпечення, програмними засобами проєктування програмного забезпечення, уніфікованою мовою проектування UML.

Проєктування програмного забезпечення – логічно складна, трудомістка і тривала робота, що вимагає високої кваліфікації задіяних фахівців. У процесі розробки і функціонування програмного забезпечення інформаційні потреби користувачів постійно змінюються або уточнюються, що ще більш ускладнює розробку і супровід таких систем.

Ці проблеми можуть бути істотно полегшені за рахунок використання сучасних структурних методів та потужних програмних засобів підтримки методології структурного аналізу.

У поняття архітектури програмного забезпечення включають структури даних, поведінку програмного забезпечення і користувача. У вітчизняній і зарубіжній літературі описана велика кількість формальних і інструментальних засобів моделювання архітектури програмного забезпечення, які відрізняються своїми властивостями, складністю використання застосованого в них математичного апарату, можливістю практичного застосування. Студент повинен вміти вибрати ефективний інструментарій для виконання безпосередньо своєї роботи – проектування інформаційних та програмних систем автоматизації.

Курс «Архітектура та проектування програмного забезпечення» включає в себе вивчення методів і сучасної комп'ютерної технології проектування програмних та інформаційних систем: аналіз взаємозв’язку сутностей, побудову діаграм класів, діаграм діяльності різного роду, генерацію відповідних структур даних та кодів на мові програмування.

В процесі вивчення дисципліни особлива увага приділяється теоретичній трактовці предмету, встановлюється зв‘язок теоретичних знань, вмінь та навичок з технологіями проектування реальних програмних застосувань, вирішенню загальних проблем, які виникають при проектуванні та розробці програмного забезпечення.

# Загальні рекомендації до виконання лабораторних робіт

Лабораторні заняття з дисципліни «Архітектура та проектування програмного забезпечення» призначені для студентів освітньо-кваліфікаційного рівня «бакалавр» спеціальності 121 «Інженерія програмного забезпечення» освітньо-професійної програми «Програмна інженерія».

При виконанні кожної роботи необхідно ознайомитися з теоретичним матеріалом та завданнями до лабораторної роботи. Захист лабораторної роботи відбувається за наступним сценарієм: студент відповідає на поставленні викладачем питання, демонструє оригінальність роботи, пояснює хід виконання лабораторної роботи. Для пояснення деяких результатів та демонстрації набутих навичок студент може скористатись комп’ютером.

Для отримання оцінки з виконання лабораторної роботи студент повинен підготувати звіт та завантажити його до СЕЗН Moodle у встановлені терміни. Звіт з виконання лабораторної роботи повинен мати:

* титульний аркуш;
* тему роботи;
* хід роботи, який позначається у виконанні її завдань;
* результати розрахунків;
* аналіз отриманих результатів виконання завдань;
* скорочені відповіді на запитання для самоперевірки.

Для завантаження до СЕЗН Moodle файл повинен бути у таких форматах, як docx, odt або pdf. Формат pdf є переважним. Якщо розмір файлу перевищує встановлені обмеження для завантаження, скористайтесь будь-яким графічним редактором для зменшення розміру наявних e ньому рисунків. Не використовуйте програмне забезпечення, що використовується для об’єднання файлів у один архівний файл. Звіт, який завантажено у архівному файлі буде вважатись непідготовленим.

# Лабораторна робота №1. Складні системи та основні принципи об'єктно-орієнтованого підходу

**Тема:** Складні системи та основні принципи об'єктно-орієнтованого підходу.

**Мета:** Освоїти основні ознаки складних систем та основні принципи об’єктно-орієнтованого підходу.

*Теоретичні відомості*

**П’ять ознак складної системи**:

1. Складні системи часто є ієрархічними та складаються з узаємозалежних підсистем, які в свою чергу також можуть бути розділені на підсистеми, і т.д., аж до „найнижчого рівня”. Архітектура системи складається як з компонентів, так і з (ієрархічних) відношень цих компонентів.
2. Вибір, які компоненти в даній системі вважаються елементарними, є відносно довільним та в великій мірі покладається на дослідника. Нижній рівень для одного користувача може виявитись достатньо високим для іншого.
3. Внутрішньокомпонентний зв’язок зазвичай є сильнішим, ніж зв’язок між компонентами. Це дозволяє відділяти „високочастотні” взаємодії всередині компонент від „низькочастотної” динаміки взаємодії між компонентами .
4. Ієрархічні системи зазвичай складаються з небагатьох типів підсистем, по-різному скомбінованих та організованих.
5. Будь-яка працююча складна система є результатом розвитку більш простої системи, що працювала раніше. Складна система, спроектована „з нуля”, ніколи не запрацює. Слід починати з працюючої простішої системи.

**Приклади складних систем:**

* «астрономічний» всесвіт;
* літак;
* диспетчерська система управління повітряним рухом;
* ринкова економіка.

**Основні принципи об’єктно-орієнтованої парадигми.**

**Інкапсуляція.** У об'єктно-орієнтованих системах дані комбінуються з конкретною поведінкою, тобто з діями, здійснюваними над ними. Все це об'єднується в клас. Даний процес називається інкапсуляцією. По-іншому її можна описати, сказавши, що ми розділяємо доданок на невеликі фрагменти пов'язаної функціональності. При цьому ми приховуємо деталі реалізації методів класу, залишаючи назовні лише взаємодію через інтерфейс.

Завдяки використанню інкапсуляції, досягаються щонайменше дві переваги:

* можливість зміни внутрішнього представлення даних екземпляра класу без необхідності зміни програмного коду клієнтів класу;
* зменшення ймовірності помилок внаслідок приведення екземпляра класу до непередбаченого стану.

Припустимо, управління банку постановило, що якщо клієнт має кредитний рахунок, цей кредит може бути використаний як овердрафт на його рахунку "до запитання".

У неінкапсульованій системі ми починаємо з вузьконаправленого аналізу змін, які необхідно внести до системи. Як правило, ми не знаємо, де в системі знаходяться всі звернення до функції зняття з рахунку, тому доводиться шукати їх скрізь. Після того, як вони знайдені, ми повинні внести до них деякі зміни, щоб реалізувати нові вимоги; Якщо ми працюємо ретельно, то, ймовірно, зможемо виявити близько 80% випадків використання даної функції.

В інкапсульованій системі не потрібно проводити такий детальний аналіз. Ми дивимося на модель системи і виявляємо, де інкапсульована відповідна поведінка (дія зняття з рахунку). Після його локалізації залишається внести необхідні поправки один раз лише в цьому об'єкті.

**Успадкування.** У об'єктно-орієнтованих системах успадкування є механізм, що дозволяє створювати нові об'єкти, ґрунтуючись на тих, що вже існують. Породжуваний (child) об'єкт-нащадок успадковує властивості батьківського (parent) об'єкту, що його породжує.

У банківській системі успадкування можна застосовувати для роботи з різними типами рахунків. Наш гіпотетичний банк обслуговує чотири типи рахунків: до запитання (checking), ощадний (savings), кредитний (credit) і депозитний сертифікат. Ці різні типи рахунків мають схожі властивості, до яких відносяться номер рахунку, ставка відсотка і власник. Отже, можна створити батьківський об'єкт account (рахунок) із загальними характеристиками всіх рахунків (рис. 1.1).



Рисунок 1.1 – Успадкування різних типів рахунків від «рахунку взагалі»

**Поліморфізм.** Поліморфізм означає наявність багатьох форм або реалізацій конкретної функціональності. В термінах об'єктно-орієнтованих систем це означає, що конкретні функціональні можливості можуть мати багато реалізацій. Ще спроба визначення: це наявність одного інтерфейсу для різних варіантів спорідненої функціональності.

Припустимо, що ми розробляємо систему малювання графіків. Коли користувач хоче щось накреслити — будь то лінія або прямокутник; система повинна виконати команду Draw (намалювати). Наша система включає багато типів різних форм, кожна з яких містить поведінку, що дозволяє їй намалювати себе. За допомогою поліморфізму система визначає, яку фігуру вона повинна намалювати.

Без поліморфізму код функції draw() виглядає приблизно таким чином:

**Function Shape.DrawMe()**

**{**

**CASE Shape.Type**

**Case “Circle”**

**Shape.drawCircle();**

**Case “Rectangle”**

**Shape.drawRectangle ();**

**Case “Line”**

**Shape.drawLine ();**

**END CASE**

**}**

Як бачимо, довелось реалізувати три інтерфейси для малювання себе.

При використанні поліморфізму код для малювання є зверненням до функції DrawMe() викреслюваного об'єкту, наприклад:

**Function draw()**

**{**

**Shape.drawMe();**

**}**

Зараз маємо лаконічніший інтерфейс. При цьому кожна форма (круг, лінія, прямокутник і так далі) повинна містити власну функцію DrawMe() для її малювання.

**Завдання до лабораторної роботи №1**

1. Навести приклади складної системи (2-3 приклади) і проаналізувати, чи задовольняє вона п'яти ознакам складних систем.
2. Навести приклади використання у розробці програмного забезпечення, або деінде трьох принципів об’єктно-орієнтованої парадигми: інкапсуляція, успадкування, поліморфізм.
3. За результатами виконання роботи оформити звіт.

**Запитання для самоперевірки**

1. П’ять ознак складної системи.
2. Програмне забезпечення (ПЗ) як складна система.
3. Яка роль декомпозиції в боротьбі зі складністю
4. Що таке алгоритмічна декомпозиція?
5. Що таке об’єктно-орієнтована декомпозиція?
6. Вибір між алгоритмічною та об’єктно-орієнтованою декомпозицією
7. Що таке інкапсуляція?
8. Що таке успадкування?
9. Що таке поліморфізм?

# Лабораторна робота №2. Формулювання призначення і цілей створення інформаційної системи

**Тема:** Формулювання призначення і цілей створення інформаційної системи.

**Мета**: Засвоїти формування розділу «Призначення і цілі створення системи» технічного завдання (ТЗ) на створення автоматизованої інформаційної системи (АІС) згідно ГОСТ 34.602-89.

*Теоретична частина*

ТЗ на АІС є основним документом, що визначає вимоги і порядок створення автоматизованої інформаційної системи, відповідно до якого проводиться розробка АІС і її приймання при введенні в дію. ТЗ на АІС розробляють на систему в цілому, призначену для роботи самостійно або у складі іншої системи.

ТЗ на АІС містить наступні розділи, які можуть бути розділені на підрозділи:

* загальні відомості;
* призначення і цілі створення(розвитку) системи;
* характеристика об'єктів автоматизації;
* вимоги до системи;
* склад і зміст робіт із створення системи;
* порядок контролю і приймання системи;
* вимоги до складу і змісту робіт з підготовки об'єкту автоматизації до введення системи в дію;
* вимоги до документування;
* джерела розробки.

У підрозділі "*Призначення системи*" вказують вид діяльності (управління, проектування і т. п.), що автоматизується, і перелік об'єктів автоматизації, на яких передбачається використати систему.

Тобто, під призначенням системи розуміємо де і що вона повинна робити.

**Приклад.** АІС “Кадри” призначена для інформаційно-аналітичного забезпечення процесів агентства “Roga’n’Kopyta” у частині виконання таких процесів:

* створення штатних розкладів;
* проведення розрахунку заробітної плати;
* оперативний облік руху кадрів;
* ведення адміністративного документообігу за персоналом та обліку праці;
* атестація та визначення потреб (навчання, підвищення кваліфікації) працівників;
* рекрутинг персоналу на вакантні посади;
* ведення архівів без обмежень строків давнини;
* публікування відкритих частин інформації системи.

У підрозділі "*Цілі створення системи*" приводять найменування і необхідні значення технічних, технологічних, виробничо-економічних або інших показників об'єкту автоматизації, які мають бути досягнуті в результаті створення АІС, і вказують критерії оцінки досягнення цілей створення системи.

**Приклад.** Нижче приведені основні цілі створення АІС “Кадри”.

* заміщення наявної інформаційної системи, яка не надає можливість комплексного інформаційно-аналітичного забезпечення процесів, перерахованих вище (у зв’язку із введенням нових нормативних правил управління кадрами);
* підвищення ефективності виконання процесів шляхом скорочення часу операції, уникнення дублювання операцій ведення даних, покращення інформаційної взаємодії учасників процесу;
* підвищення якості прийняття управлінських рішень за рахунок оперативності представлення, повноти, достовірності та зручності форматів відображення інформації;
* підвищення інформаційної відкритості та прозорості діяльності органів агентства “Roga’n’Kopyta”, підвищення зручності та комфорту (зниження фінансових та часових витрат) фізичних та юридичних осіб при отриманні інформації про кадрові потреби агентства.

Для реалізації поставлених цілей система має вирішувати такі задачі:

* оперативний облік кадрів;
* облік праці;
* розрахунок заробітної плати;
* побудова аналітичних звітів та виписок;
* інтеграцію із існуючими АІС агентства;
* створення сайту для відображення відкритих частин інформації.

**Завдання до лабораторної роботи №2**

1. Згідно з вибраною предметною областю визначити наступні підрозділи з розділу "*Призначення і цілі створення системи*" технічного завдання:

* призначення системи;
* цілі створення системи;
* задачі, які система має вирішувати.

1. За результатами виконання роботи оформити звіт.

**Запитання для самоперевірки**

1. Що таке ціль створення системи і що таке призначення системи?
2. Що таке уніфікований процес і як його можна узгодити з процесом створення інформаційної системи, передбаченим у ГОСТах?

# Лабораторна робота №3. Визначення переліку акторів та прецедентів

**Тема:** Визначення переліку акторів та прецедентів.

**Мета**: Визначити перелік акторів та вивити всі варіанти використання системи.

*Теоретична частина*

**Прецедент** (**варіант використання**) — це опис на "високому рівні" фрагмента функціональності, яку забезпечує система. Інакше кажучи, варіант використання ілюструє, як можна використовувати систему. Наприклад, автоматичний банкомат (АТМ) надає клієнту деякий базовий набір функціональних можливостей:

* знімати гроші;
* робити вклад;
* перекладати гроші з одного рахунку на інший;
* переглядати свій баланс;
* змінювати ідентифікаційний номер;
* або виконувати оплату за безготівковим розрахунком з кредитної картки.

Кожна з описаних транзакцій є способом, яким клієнт може використовувати систему. Тому кожний з них – самостійний варіант використання.

Як виявити варіанти використання?

* вивчити відповідні документи замовника;
* врахувати думку кожної зацікавленої особи.

Для цього можна задати їм наступні питання:

* що він хоче робити з системою, що вводити, отримувати?
* яка інформація щодо зовнішніх подій потрібна системі?
* яка інформація щодо внутрішнього стану системи потрібна користувачу?

Набір варіантів використання надає можливість побачити систему в цілому на найвищому рівні. Тому варіантів використання не повинно бути надто багато, щоб клієнту не довелося довго шукати в документації, що робитиме система. В той же час варіантів використання повинно бути достатньо для повного опису поведінки системи. Модель типової системи зазвичай містить від 20 до 50 варіантів використання.

Щоби пересвідчитись, що виявлені всі варіанти використання, слід відповісти на питання:

* чи присутня кожна функціональна вимога хоч би в одному варіанті використання?
* чи враховано, як з системою буде працювати кожна зацікавлена особа?
* чи враховані всі зовнішні системи, з якими взаємодіятиме нова система?

**Актори** діляться на три основні типи: *користувачі системи*, *інші системи*, що взаємодіють з даною, і *час*.

*Користувачі системи* — це фізичні особи. Вони найбільш типові і є практично в кожній системі. У системі АТМ, наприклад, до цього типа відносяться клієнти і обслуговуючий персонал. Називаючи дійових осіб, краще використовувати їх ролеві імена, а не посади. Напр., певний працівник, який відповідає за підтримку сервера банкоматів (нехай це буде роль), в обід зніме гроші і, таким чином, виступить у ролі користувача.

Другим типом акторів є *інша система*. Припустимо, що в банку є кредитна система для роботи з інформацією про кредитні рахунки клієнтів. Наша система АТМ повинна мати можливість взаємодіяти з кредитною системою, тоді остання стає актором.

Третій найбільш поширений тип актора — *час*. Час стає дійовою особою, якщо від нього залежить запуск певних подій у системі. Наприклад, система АТМ може кожного дня опівночі виконувати певні службові процедури.

Актори знаходяться поза сферою дії того, що ми розробляємо, і, отже, не підлягають контролю з нашого боку. Якщо ми збираємося змінювати або розробляти і кредитну систему також, то вона потрапляє в наш проект і, таким чином, не може бути показана як актор.

**Приклад.** Розглянемо визначення переліку акторів та прецедентів на прикладі предметної області *Технічна служба автотранспортного підприємства*.

Головною функцією Технічної служби (ТС) автотранспортного підприємства (АТП) є підтримка рухомого складу АТП у справному стані із мінімальними витратами на технічне обслуговування і ремонт автотранспорту. Мінімізація витрат передбачає ефективне використання запасних частин, як нових, так і тих, що вже були в експлуатації.

На рис. 3.1 наведена структура каталогу деталей марки автомобіля автозаводу-виготовлювача.



Рисунок 3.1 – Структурна формула коду деталі у Каталозі деталей

Рух деталей по підприємству в процесі їхнього використання можна представити такою схемою (див. рис. 3.1):



Накладна зовнішнього надходження

Цехова накладна

Вимога

Дефектна відомість

Рисунок 3.2 – Рух деталей по підприємству

Форми документообігу ТС відображають такі технологічні процеси:

* Надходження від постачальників нових запчастин на центральний склад (ЦС) в підзвіт комірника ЦС; ця операція фіксується комірником ЦС за допомогою первинного документу (ПД) *Накладна зовнішнього надходженнь;*
* Поповнення запасних частин за рахунок виготовлення деталей і здачі їх на Центральний склад в підзвіт комірника ЦС на основі ПД *Цехова накладна*;
* Поповнення на перехідному складі (ПС) в підзвіт комірника ПС і подальше використання деталей, що вже були в експлуатації. Вони виникають при дефектуванні деталей списаних вузлів та агрегатів автомобілів на основі ПД *Дефектна відомість,* яку формує інженер ТС;
* Якщо автомобіль ремонтується, визначення інженером ТС деталей вузла чи агрегату автомобіля, які необхідно замінити на нові, в ПД *Дефектна відомість*, що створюється при розподілі деталей вузла чи агрегату автомобіля на справні і несправні. Несправні деталі списуються у брухт. Ця операція має назву *дефектування*;
* Якщо автомобіль списується, визначення інженером ТС справних деталей вузла чи агрегату автомобіля, які необхідно оприбуткувати на ПС, та несправних деталей, які списуються у брухт. Результат також фіксується в ПД *Дефектна відомість;*
* Визначення інженером ТС наявності деталі на складі шляхом видачі відповідної *Відомості наявності*;
* Отримання інженером ТС деталі зі складу для ремонту автомобіля на основі ПД *Вимога*.

У практиці управління АТП створення окремої БД для технічної служби недоцільне, оскільки перелічена інформація в повному обсязі потрібна для бухгалтерського обліку і знаходиться в базі даних інтегрованої АСУ підприємства (ERP-системи). Цей перелік документів технічної служби буде тільки фрагментом БД ПО обліку і управління в АТП.

Визначимо перелік акторів.

Як випливає з опису предметної області (ПО), в ній діють 3 актора:

* Комірник ЦС;
* Комірник ПС;
* Інженер ТС.

Визначимо перелік прецедентів для кожного актора та результати представимо у таблиці 3.1.

Таблиця 3.1 – Перелік акторів та прецедентів

|  |  |
| --- | --- |
| *Актор* | *Прецедент* |
| Комірник ЦС | * Введення ПД *Накладна зовнішнього надходження* * Введення ПД *Цехова накладна* * Введення ПД *Вимога* * Видача звіту *Відомість наявності* |
| Комірник ПС | * Введення ПД *Вимога* * Видача звіту *Відомість наявності* |
| Інженер ТС | * Введення ПД *Дефектна відомість* в разі ремонту автомобіля * Введення ПД *Дефектна відомість* в разі списання автомобіля * Видача звіту *Відомість наявності* |

**Завдання до лабораторної роботи №3**

1. Освоїти предметну область *Технічна служба автотранспортного підприємства*, яка є спільним для всіх прикладом.
2. Освоїти предметну область згідно свого варіанту.
3. Для власного варіанту предметної області:

* визначити перелік акторів (рекомендація - до 4-х);
* визначити варіанти використання (прецеденти) в кількості до 10.

1. Перелік акторів та прецеденти представити у вигляді таблиці 3.1.
2. За результатами роботи оформити звіт.

**Запитання для самоперевірки**

1. Що таке прецедент?
2. Хто такі актори?
3. На які основні типи діляться актори?
4. Які деталі зберігаються на Центральному складі ТС та які – на Перехідному?
5. Якими типами ПД обліковується рух деталей по складах і ТС?

# Лабораторна робота №4. Опис прецедентів

**Тема:** Опис прецедентів.

**Мета**: Виконати документування прецедентів.

*Теоретична частина*

Для документування варіанту реалізації рекомендують скласти документ *Потік подій.* Метою потоку подій є документування процесу обробки даних, що реалізується в рамках варіанту використання. Цей документ детально описує, що робитимуть користувачі системи і що — сама система. Потік подій також не повинен залежати від реалізації.

Зазвичай потік подій містить:

* Короткий опис.
* Передумови (pre-conditions).
* Основний потік подій.
* Альтернативний потік подій.
* Постумови (post-conditions).

Розглянемо послідовно ці складові частини.

***Опис***

Кожен варіант використання повинен мати пов'язаний з ним короткий опис того, що він робитиме. Наприклад, варіант використання "Перевести гроші" системи АТМ може містити наступний опис:

*Варіант використання "Перевести гроші" дозволяє клієнтові або службовцеві банку перекладати гроші з одного рахунку до запитання або ощадного рахунку на інший.*

Опис має визначати типи задіяних користувачів та очікуваний ними результат.

***Передумови***

Передумови варіанту використання — це такі умови, які мають бути виконані, перш ніж варіант використання почне свою роботу. Наприклад, такою умовою може бути виконання іншого варіанту використання або наявність у користувача прав доступу, потрібних для запуску даного варіанту використання. Не у всіх варіантів використання бувають попередні умови.

***Основний та альтернативний потоки подій***

Конкретні деталі варіантів використання відбиваються в основному в альтернативному потоках подій. Потік подій поетапно описує, що повинне відбуватися під час виконання закладеної у варіанти використання функціональності. Потік подій приділяє увагу тому, що (а не як) робитиме система, причому описує це з точки зору користувача.

Наприклад, потік подій варіанту використання "Зняти гроші" може виглядати таким чином:

*Основний потік*

1. Варіант використання починається, коли клієнт вставляє свою картку в АТМ.
2. АТМ видає вітання і пропонує клієнтові ввести свій персональний ідентифікаційний номер.
3. Клієнт вводить номер.
4. АТМ підтверджує введений номер. Якщо номер не підтверджується, виконується альтернативний потік подій А1.
5. АТМ виводить список доступних дій:
   * Покласти гроші на рахунок.
   * Зняти гроші з рахунку.
   * Перевести гроші.
6. Клієнт вибирає пункт "Зняти гроші".
7. АТМ запрошує, скільки грошей потрібно зняти.
8. Клієнт вводить необхідну суму.
9. АТМ визначає, чи досить на рахунку грошей. Якщо грошей недостатньо, виконується альтернативний потік А2.
10. АТМ віднімає необхідну суму з рахунку клієнта.
11. АТМ видає клієнтові необхідну суму готівкою.
12. АТМ повертає клієнтові його картку.
13. Варіант використання завершується.

*Альтернативний потік А1: введення неправильного ідентифікаційного номера*

1. АТМ інформує клієнта, що ідентифікаційний номер введений неправильно.
2. АТМ повертає клієнтові його картку.
3. Варіант використання завершується.

*Альтернативний потік А2: недостатньо грошей на рахунку*

1. АТМ інформує клієнта, що грошей на його рахунку недостатньо.
2. АТМ повертає клієнту його картку.
3. Варіант використання завершується.

**Завдання до лабораторної роботи №4**

1. Освоїти предметну область *Технічна служба автотранспортного підприємства*, яка є спільним для всіх прикладом.
2. Освоїти предметну область згідно свого варіанту.
3. Для власного варіанту предметної області:

* визначити головні варіанти використання (рекомендація - до 4-х);
* задокументувати обрані варіанти використання.

1. За результатами роботи оформити звіт.

**Запитання для самоперевірки**

1. Програмний продукт Rational Rose.
2. Елементи екрану Rose.
3. Браузер Rose.
4. Документування елементів моделі Rose.

# Лабораторна робота №5. Створення діаграми прецедентів

**Тема:** Створення діаграми прецедентів.

**Мета:** Відобразити взаємодію акторів і прецедентів у діаграмі прецедентів.

*Теоретичні відомості*

Розглянемо відображення взаємодії акторів і прецедентів у діаграмі прецедентів на прикладі предметної області *Технічна служба автотранспортного підприємства (ТС АТП)*. Перелік акторів та прецедентів був виявлений нами у лабораторній роботі №3 і представлений у таблиці 5.1.

Таблиця 5.1 – Перелік акторів та прецедентів

|  |  |
| --- | --- |
| *Актор* | *Прецедент* |
| Комірник ЦС | * Введення ПД *Накладна зовнішнього надходження* * Введення ПД *Цехова накладна* * Введення ПД *Вимога* * Видача звіту *Відомість наявності* |
| Комірник ПС | * Введення ПД *Вимога* * Видача звіту *Відомість наявності* |
| Інженер ТС | * Введення ПД *Дефектна відомість* в разі ремонту автомобіля * Введення ПД *Дефектна відомість* в разі списання автомобіля * Видача звіту *Відомість наявності* |

Для створення діаграми прецедентів у програмі *Rational Rose* потрібно виконати наступні дії:

1. Запускаємо програму *Rational Rose Enterprise Edition*. Зачиняємо вікно *Create New Model* кнопкою *Cancel*, а також вікно *Class diagram*.
2. У браузері відчинимо папку *Use Case View* та перейменуємо діаграму *Main* типу *Use Case* (див. рис. 5.1). Надамо діаграмі ім’я *TechService\_UseCase*.
3. Відчиняємо діаграму – клікаємо правою кнопкою миші на назві діаграми і в контекстному меню обираємо *Open* (див. рис. 5.2). Відчиняється пусте вікно діаграми.
4. Додавати елементи на діаграму можна двома способами (див. рис. 5.3):

* За допомогою меню *Tools\Create*. Пропонуються для створення всі можливі елементи;
* За допомогою панелі діаграми – найбільш зручний спосіб. Пропонуються найбільш вживані елементи, яких нам цілком вистачить.

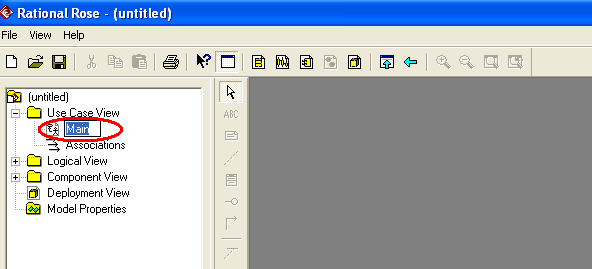


Рисунок 5.1 – Перейменування діаграми *Use Case*

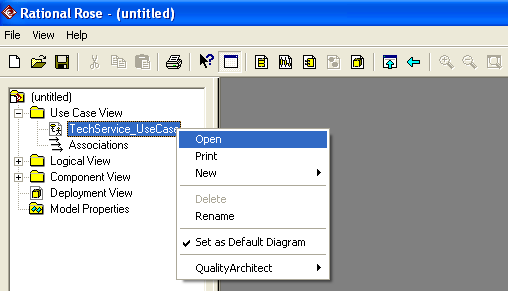


Рисунок 5.2 – Відчинення діаграми *Use Case*

1. Створимо на діаграмі акторів з таблиці 5.1. Для цього треба на панелі діаграми натиснути кнопку актора, після цього клікнути в потрібному місці на діаграмі, затим дати актору ім’я (див. рис. 5.4).
2. В разі необхідності перейменування актора або його документування можна відчинити специфікацію актора подвійним кліком на ньому або контекстним меню (див. рис. 5.5). Документуємо акторів шляхом введення у вікно специфікації *Documentation* їхніх назв без скорочень. Для *Rational Rose* актор є класом зі стереотипом актора, який утім, можна змінити. Тому вікно специфікації актора є вікном специфікації класу.
3. Створимо прецеденти згідно таблиці 5.1. Виявляється, з 9 прецедентів 2 використовуються повторно (Введення ПД *Вимога*, Видача *Відомості наявності*), тому на діаграмі їх усього 6. Пов’яжемо акторів та прецеденти суцільною стрілкою *Unidirectional Association* (третя знизу на панелі діаграми). Користуємося правилом: стрілка йде від активного елемента (в цьому разі актора), навіть для прецеденту видачі відомості наявності, оскільки актор ініціює прецедент (див. рис. 5.6).

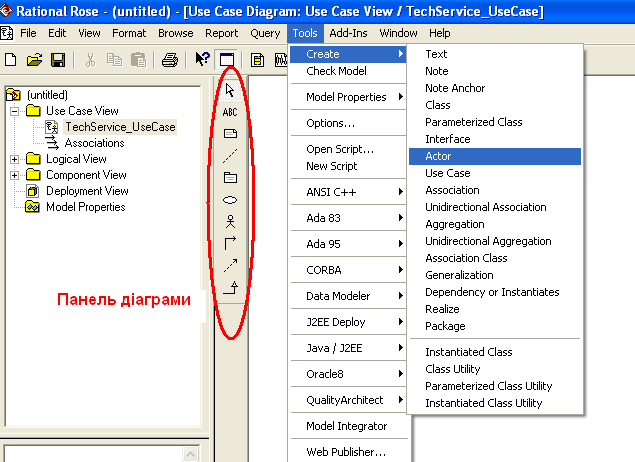


Рисунок 5.3 – Додавання елементів до діаграми *Use Case*

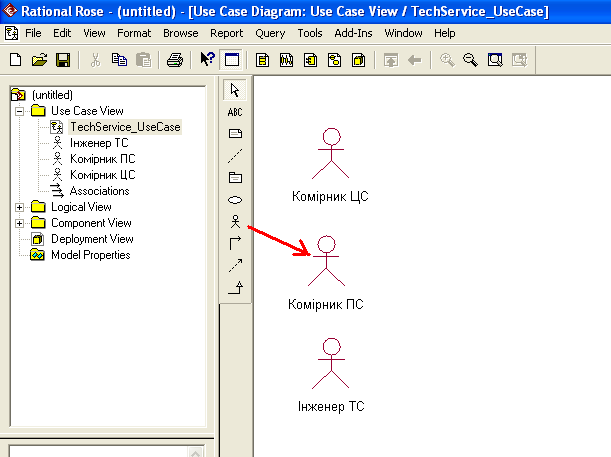


Рисунок 5.4 – Додавання акторів до діаграми *Use Case*

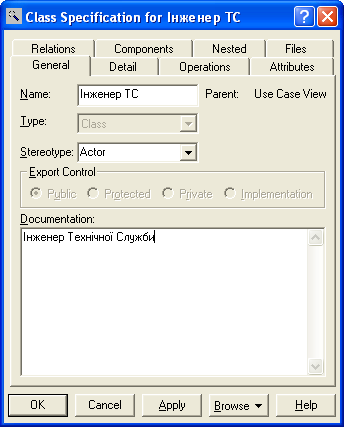


Рисунок 5.5 – Вікно специфікації актора



Рисунок 5.6 – Додавання асоціацій від акторів до прецедентів

1. Врахуємо, що всі прецеденти введення ПД будь-якого типу повинні мати багато спільного (незалежно від реалізації), тому вони мають узагальнюватись у базовому прецеденті *Введення ПД.* Додамо цей прецедент та зв’язки узагальнення (остання кнопка на панелі діаграми) з цим прецедентом (див. рис. 5.7).
2. При видачі *Відомості наявності* суттєвою є роль сервера БД, який готує інформацію по наявності деталей на складі. Введемо *Сервер БД* як окремий клас і позначимо звертання до нього. Дамо йому стереотип *Domain* (див. рис. 5.8).



Рисунок 5.7 – Додавання зв’язків узагальнення для прецедентів *Введення ПД*



Рисунок 5.8 – Додавання класу *Сервер БД* до діаграми

На жаль, за браком місця неможливо вивести ролі зв’язків-асоціацій („вводить”, „видає”, „звертається по дані”). При наявності місця ролі можна записати у полі назви зв’язку специфікації асоціації.

1. Прецедент *Вхід у систему* включений у прецеденти *Введення ПД, Видача Відомості наявності.* Також він звертається до *Сервера БД* для встановлення прав користувача. Створимо прецедент і відповідні зв’язки. Зв’язок включення – штрихова стрілка, друга знизу на панелі діаграми (див. рис. 5.9).



Рисунок 5.9 – Діаграма після додавання прецеденту *Вхід у систему*

1. Збережемо модель у mdl-файлі, бажано на власній флешці. Це робить кнопка з дискетою на стандартній панелі інструментів.
2. При необхідності вставити діаграму в *Microsoft Word* треба виділити всі її елементи (Ctrl+A), скопіювати в буфер (Ctrl+C) та вставити у *Microsoft Word* (Ctrl+V).

**Завдання до лабораторної роботи №5**

Використовуючи перелік акторів та прецедентів для своєї предметної області з лабораторної роботи №3:

1. Створити в програмі *Rational Rose* діаграму прецедентів.
2. Додати на діаграму акторів.
3. Додати на діаграму прецеденти.
4. Встановити зв’язки між акторами та прецедентами.
5. Встановити ролі зв’язків-асоціацій між акторами та прецедентами.
6. За результатами виконання роботи оформити звіт.

**Запитання для самоперевірки**

1. Хто є користувачами діаграми прецедентів?
2. Чим актор відрізняється від об’єктів інших класів?
3. Які 3 види зв’язків відображені на діаграмі рис.5.9?
4. Що таке відношення узагальнення? Наведіть приклади.
5. Що таке відношення включення? Наведіть приклади.

# Лабораторна робота №6. Створення діаграми послідовності

**Тема:** Створення діаграми послідовності.

**Мета:** Відобразити основні та альтернативні потоки подій на діаграмі послідовності.

*Теоретичні відомості*

**Визначення основного та альтернативних потоків подій.**

Оберемо прецедент середньої складності *Введення ПД "Дефектна відомість" для ремонту авто.*

Потоки подій такі:

*1. Введення ПД "Дефектна відомість" для ремонту авто:*

1). Вхід інженера ТС в систему.Він вводить логін і пароль.

2) Пароль перевіряється на сервері. Якщо він невірний, виконується альтернативний сценарій "Невірний пароль".

3) Інженер ТС заходить у папку ПД "Дефектні відомості", створює новий ПД

4) Новий ПД створюється в таблиці ПД в БД.

5) Інженер ТС заповнює реквізити ПД (№, дата, підстава).

6) Реквізити ПД зберігаються на сервері .

7) Інженер ТС вводить рядки з номенклатурою і кількістю списаних деталей.

8) Дані про деталі зберігаються в БД.

9) Інженер ТС вводить кількість оприбуткованого брухту.

10) Дані про брухт зберігаються в БД.

11) Інженер ТС проводить ПД.

12) Дані про рух деталей, бухгалтерські проводки і зміни залишків зберігаються в БД.

*2. Невірний пароль:*

1) Виводиться повідомлення "Невірний пароль".

2) Інженер ТС вводить логін і пароль. Це дозволено до 3-х разів.

3) Якщо 3 спроби використано, вихід.

4) Якщо логін і пароль вірні, продовжуємо основний потік подій з п.3.

Вставимо описи потоків подій у поле *Documentation* специфікації прецеденту (див. рис. 6.1).

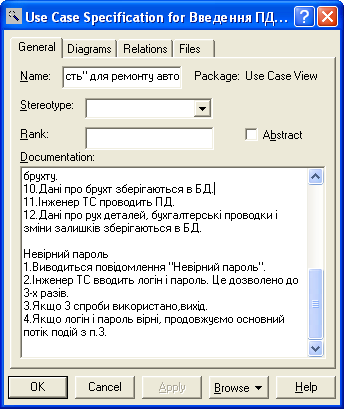


Рисунок 6.1 – Специфікація прецеденту з описом потоків подій

**Визначення класів, включаючи актора, які беруть участь у потоках подій прецеденту.**

Таблиця 6.1 – Класи, задіяні в прецеденті *Введення ПД "Дефектна відомість" для ремонту авто*

|  |  |
| --- | --- |
| *Клас* | *Прецедент* |
| Інженер ТС | Його функції визначені в попередній лабораторній роботі |
| Задача *Облік матеріальних цінностей* | Це доданок на клієнтській робочій станції. Подрібніше – в діаграмі компонентів. |
| Сервер БД | Обслуговує кілька робочих станцій, де обслуговуються по кілька задач |

**Відображення взаємодії класів у діаграмі послідовності.**

На цьому етапі, керуючись описом потоків подій, розробимо в *Rational Rose* дві діаграми послідовності для обраного прецеденту.

1. Запускаємо програму *Rational Rose Enterprise Edition*. Зачиняємо вікно *Create New Model* кнопкою *Cancel*, а також вікно *Class diagram*.
2. Відчинимо файл моделі технічної служби *TechService.mdl*.
3. У браузері відчинимо папку *Use Case View* та створимо нову діаграму послідовності, (див. рис. 6.2). Надамо діаграмі ім’я *TechService\_Seq*.
4. Відчиняємо діаграму: клікаємо правою кнопкою миші на назві діаграми і в контекстному меню обираємо *Open*. Відчиняється пусте вікно діаграми (див. рис. 6.3).

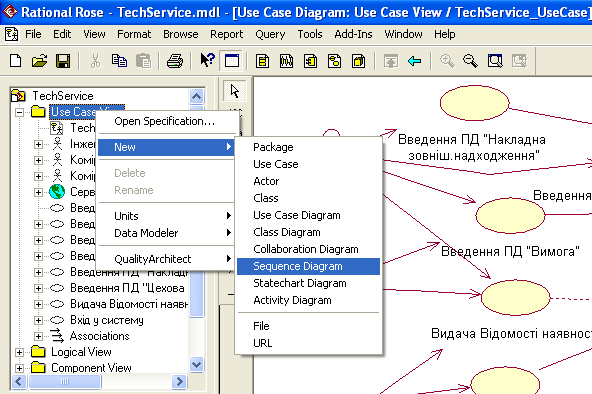


Рисунок 6.2 – Створення нової діаграми послідовності

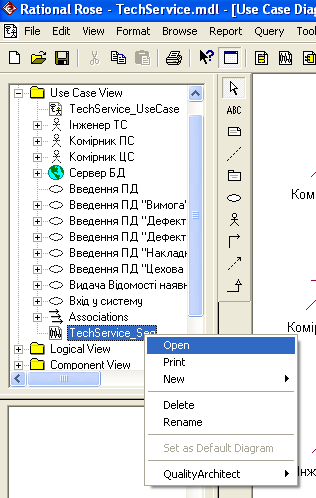


Рисунок 6.3 – Відчинення діаграми послідовності

1. Додавати елементи на діаграму можна двома способами:

* За допомогою меню *Tools\Create*. Пропонуються для створення всі можливі елементи;
* За допомогою панелі діаграми – найбільш зручний спосіб. Пропонуються найбільш вживані елементи, яких нам цілком вистачить.

Як передбачено діаграмою, розташуємо актора і класи у верхній частині діаграми (див. рис. 6.4).

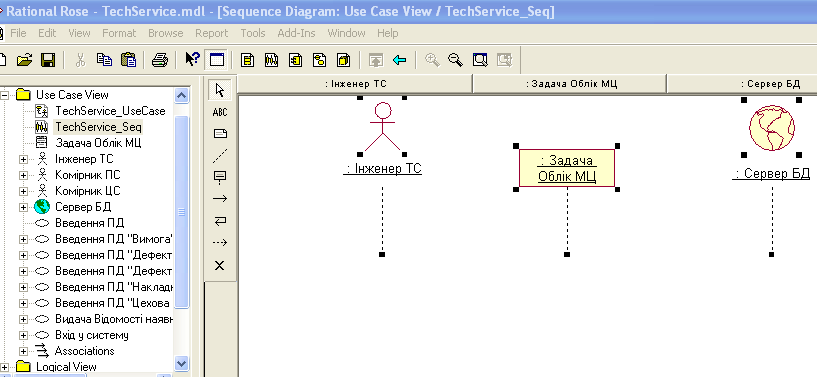


Рисунок 6.4 – Додавання класів до діаграми послідовності

1. Аналізуючи потоки подій, доповнимо специфікації класів операціями, по які до них звертаються інші класи.

Вийшли наступні операції (див. рис. 6.5, 6.6).

1. Відобразимо на діаграмі повідомлення (тобто звертання класів до інших класів) в хронологічному порядку. Порядок дій:

* Обрати на панелі діаграми горизонтальну стрілку *Object Message;*
* Провести мишею від лінії життя відправника повідомлення до лінії життя отримувача;

За допомогою правої кнопки миші відчиняється контекстне меню, де можна обрати відповідну операцію класу – повідомлення.

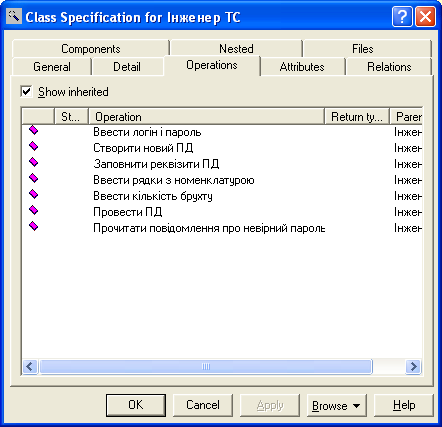


Рисунок 6.5 – Операції *Інженера ТС*

Операція задачі *Облік МЦ –* лише *Вихід.*

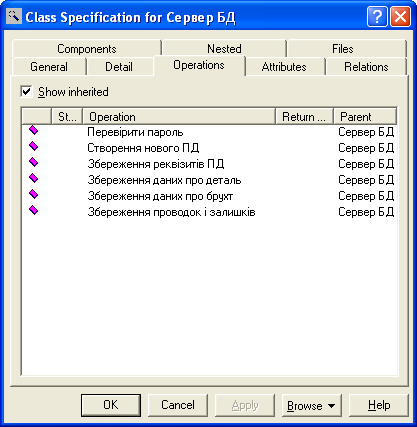


Рисунок 6.6 – Операції класу *Сервер БД*

В результаті отримали діаграму послідовності для прецеденту *Введення ПД "Дефектна відомість" для ремонту авто* (див. рис. 6.7).



Рисунок 6.7 *–* Діаграма послідовності для прецеденту *Введення ПД "Дефектна відомість" для ремонту авто*

1. Так само заповнимо діаграму для альтернативного потоку подій *Невірний пароль* (див. рис. 6.8).



Рисунок 6.8 – Діаграма послідовності для прецеденту *Невірний пароль*

**Відображення взаємодії класів у діаграмі послідовності.**

За допомогою пункту меню *Browse\Create collaboration diagram* створимо діаграму кооперації для нашого прецеденту (див. рис. 6.9).



Рисунок 6.9 – Діаграма кооперації для прецеденту *Введення ПД "Дефектна відомість" для ремонту авто*

1. Збережемо модель у mdl-файлі, бажано на власній флешці. Це робить кнопка з дискетою на стандартній панелі інструментів.
2. При необхідності вставити діаграму в *Microsoft* *Word* треба виділити всі її елементи (Ctrl+A), скопіювати в буфер (Ctrl+C) та вставити у *Microsoft* *Word* (Ctrl+V).

**Завдання до лабораторної роботи №6**

Освоїти предметну область *Технічна служба автотранспортного підприємства*, яка є спільним для всіх прикладом. Освоїти предметну область згідно свого варіанту.

Використовуючи діаграму прецедентів з лабораторної роботи №4 для власного варіанту предметної області:

1. Визначити основний та альтернативні потоки подій для одного з прецедентів (рекомендація - до 3-х).
2. Визначити класи, включаючи акторів, які беруть участь у потоках подій, в кількості до 5.
3. Відобразити взаємодію класів у діаграмі послідовності для своєї предметної області.
4. З готової діаграми послідовності створити діаграму кооперації.
5. За результатами виконання роботи оформити звіт.

**Запитання для самоперевірки**

1. Хто є користувачами діаграми послідовності?
2. Що таке основний і альтернативні потоки подій?
3. Що таке лінія життя об’єкту?
4. В якому напрямку іде стрілка *Object Message* при зображенні передачі повідомлення між об’єктами?
5. Як прив’язати скрипт до повідомлення? Навіщо це потрібно? Наведіть приклади.
6. Як створити діаграму кооперації по готовій діаграмі послідовності?

# Лабораторна робота №7. Створення діаграми класів

**Тема:** Створення діаграми класів.

**Мета:** Відобразити властивості, зв’язки та ієрархію класів на діаграмі класів.

*Теоретична частина*

**Створення діаграми класів для відображення ієрархії успадкування.**

Як і в лабораторній роботі №5, розглянемо прецедент *Введення ПД "Дефектна відомість" для ремонту авто*.

1. Створимо нову діаграму класів. Для цього зручно відчинити контекстне меню на представленні *Logical View* (див. рис. 7.1).

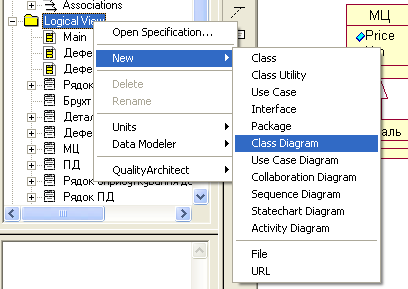


Рисунок 7.1 – Створення нової діаграми класів

1. Створимо або виведемо на діаграму класів необхідні класи для показу їхнього успадкування. Відобразимо 3 ієрархії успадкування, які стосуються прецеденту:

* Типи первинних документів (ПД);
* Типи рядків ПД;
* Види матеріальних цінностей (МЦ).

Якщо клас з ієрархії успадкування вже існує (у нас таких нема), треба затягнути його на діаграму з браузера.

Якщо класу не існує, треба його створити тим же локальним меню, або скористатись меню *Tools\Create\Class*, або (найзручніший спосіб) обравши клас на панелі діаграми, клікнути на полі діаграми в потрібному місці. В нашому прикладі довелося створити всі класи.

1. Встановимо зв’язки узагальнення (зворотні до успадкування) між класами одної ієрархії. Для цього на панелі діаграми слід обрати стрілку *Generalization* та протягнути її від дочірнього класу до батьківського (див. рис. 7.2).

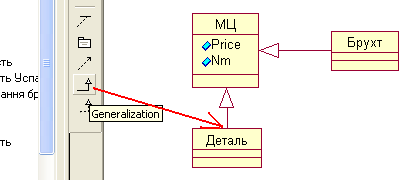


Рисунок 7.2 – Встановлення зв’язку узагальнення між класами

1. Визначимо і введемо атрибути і функції, які належать до батьківських класів. Видимість усіх цих атрибутів установлюємо *Public*. Це зручніше зробити за допомогою контекстного меню в полі переліку атрибутів (або, відповідно, операцій) специфікації класу (див. рис. 7.3).

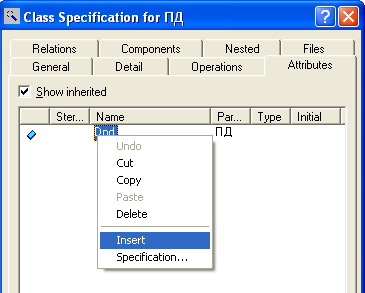


Рисунок 7.3 – Додавання атрибуту до класу

1. Оцінимо, чи є батьківські класи абстрактними. Якщо реалізація операцій однакова для різних дочірніх класів і тому міститься в батьківському класі, цей батьківський клас не є абстрактним. В іншому разі він таким є, ставимо відповідну галочку, і відтепер назва класу виводиться курсивом (див. рис. 7.4).

Інші деталі класів залишаємо для другої діаграми класів, де будуть показані зв’язки класів крім узагальнення.

В підсумку маємо наступну діаграму, яка відображає зв’язки узагальнення класів (див. рис. 7.5).

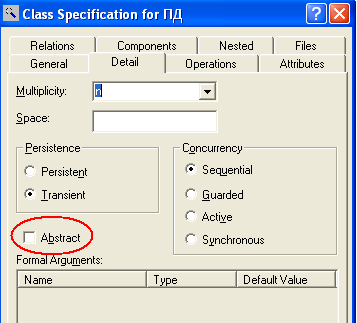


Рисунок 7.4 – Встановлення властивості класу *Абстрактний*



Рисунок 7.5 – Діаграма класів для відображення успадкування та узагальнення

**Створення діаграми класів для відображення усіх властивостей та зв’язків класів.**

При створенні цієї діаграми ми намагаємося на невеликій кількості класів (бажано до 7) показати зв’язки різних видів.

1. Створимо ще одну діаграму класів та виведемо на ній класи, при необхідності створимо нові класи.
2. Створимо зв’язки. В цьому прикладі класи типів рядків є класами-асоціаціями, які в свою чергу асоціюються з нащадками класу МЦ. В однонаправленої асоціації (див. рис. 7.6) стрілка вказує на клас, множинність якого 1. Ненаправлена асоціація створюється для зв’язків *один до одного* та *багато до багатьох* (за допомогою меню *Tools\Create\Association*).

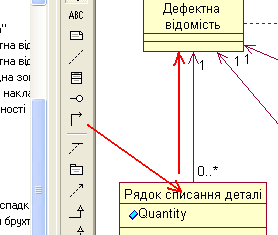


Рисунок 7.6 – Послідовність створення однонаправленої асоціації

1. Вкажемо множинність кожного зв’язку шляхом виклику контекстного меню біля кінця зв’язку і вказавши значення *Multiplicity* (див. рис. 7.7). Спробуємо пояснити множинність зв’язку на прикладі зв’язку від класу *Рядок списання деталі* (множинність 0..\*) до класу *Дефектна відомість* (множинність 1).

Таблиця 7.1 – Визначення множинність зв’язку між класами

|  |  |  |
| --- | --- | --- |
| *Питання для визначення множинності* | *Відповідь* | *Позначення* |
| Скільки дефектних відомостей може існувати для одного рядка списання деталі? | Лише одна. | 1 |
| Скільки рядків списання деталі може існувати у дефектній відомості? | Від нуля до багатьох. | 0..\* |

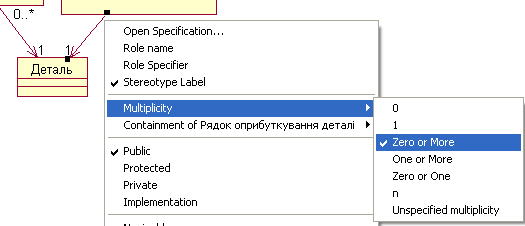


Рисунок 7.7 – Встановлення множинності одної сторони зв’язку

Після цього діаграма має наступний вигляд (див. рис. 7.8).



Рисунок 7.8 – Діаграма класів зі зв’язками асоціації

1. Для виводу зв’язку залежності відобразимо на діаграмі клас *Сервер БД*. Клас *Дефектна відомість* звертається до серверних збережених процедур, зокрема, при проведенні ПД. Тому проводимо зв’язок залежності від ПД до сервера.
2. Аналізуючи можливість включення в діаграму вкладених класів, класу інтерфейсу, зв’язків реалізації та агрегації, доходимо висновку, що вони в моделі відсутні.
3. Після встановлення зв’язків для кожного класу по можливості треба визначити в його специфікації:

* Видимість класу;
* Множинність класу;
* Стійкість класу;
* Паралелізм класу;
* Чи є клас абстрактним.

Остаточно діаграма класів має наступний вигляд (див. рис. 7.9).



Рисунок 7.9 – Остаточний вигляд діаграми класів

**Завдання до лабораторної роботи №7**

Для власного варіанту предметної області:

1. Відобразити ієрархію успадкування для класів, які беруть участь у прецеденті, розглянутому Вами в Лабораторній роботі 5 *Створення діаграми послідовності,* в кількості до 5. Вказати успадковані та власні атрибути і операції. Вказати абстрактні класи.
2. Відобразити взаємодію класів у діаграмі класів для обраного прецеденту.
3. За результатами виконання роботи оформити звіт.

**Запитання для самоперевірки**

1. Хто є користувачами діаграми класів?
2. Чим класи на діаграмі класів можуть відрізнятись від класів на діаграмі послідовності?
3. Які дії виконує Rose на підставі діаграми класів?
4. В якому напрямку іде стрілка у зв’язку однонаправленої асоціації?
5. Як визначити множинність однонаправленої асоціації?
6. В якому напрямку іде стрілка у зв’язку узагальнення?
7. Що таке стійкість класу, множинність класу, видимість класу?

# Лабораторна робота №8. Створення діаграми діяльності

**Тема:** Створення діаграми діяльності.

**Мета**: Відобразити діаграму діяльності.

*Теоретична частина*

**Приклад виконання завдання для предметної області „Технічна служба АТП”.**

Використаємо прецедент, який було розглянуто як приклад на діаграмі послідовності (лабораторна робота 6) – *Введення ПД "Дефектна відомість" для ремонту авто*. Бажано, щоби в прецеденті взаємодіяло більше одної ролі.

В контекстному меню вузла *Use Case View* створимо нову діаграму діяльності (див. рис. 8.1). Відчинимо її (перший пункт контекстного меню).

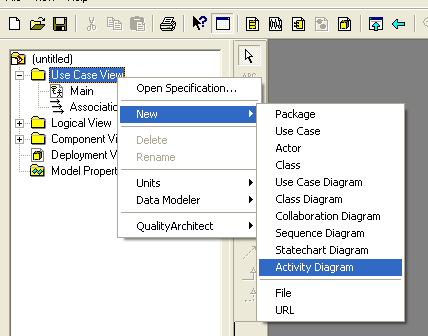


Рисунок 8.1 – Створення нової діаграми діяльності

**Доріжки.**

(Плавальні) доріжки – це організаційні одиниці або ролі, які відповідають за дії, розташовані на відповідній доріжці. У нашому прикладі є один користувач, який вводить *ПД Дефектна відомість*. Визначимо дві доріжки:

* Користувач – *інженер ТС*; виконує дії на рівні користувача – введення ПД;
* *Задача* *Облік МЦ*, що виконує дії на рівні бази даних – створення і коректування записів про ПД і різні залишки.

Для такого рівня розгляду роль СУБД, яка безпосередньо управляє записами БД, недоцільно відділяти від ролі задачі *Облік МЦ*.

Для визначення нової доріжки клікнемо на останній кнопці панелі інструментів, а потім клікнемо на полі діаграми. У капелюсі доріжки напишемо назву ролі: *Інженер ТС* або *Задача „Облік МЦ”* (див. рис. 8.2).

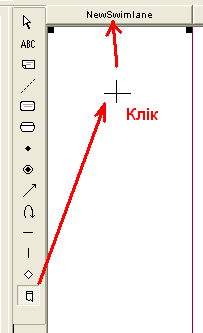


Рисунок 8.2 – Створення нової доріжки

**Створення нових дій та їхніх з’єднань.**

Інструменти на панелі діаграми діяльності описані в лекції в таблиці *Інструменти діаграми діяльності*. Створення нових елементів діаграми, як-от станів, дій, блоків рішення, синхронізації, виконується як і створення доріжок:

* Треба клікнути на відповідному інструменті на панелі інструментів,
* Після чого клікнути на полі діаграми в тому місці, де ви хочете створити елемент діаграми.

При створенні переходів треба клікнути на інструменті *Transition*, після чого тягнути стрілку від елемента-відправника до елемента-отримувача. Можна вигнути стрілку, потягнувши за неї в сторону (див. рис. 8.3).

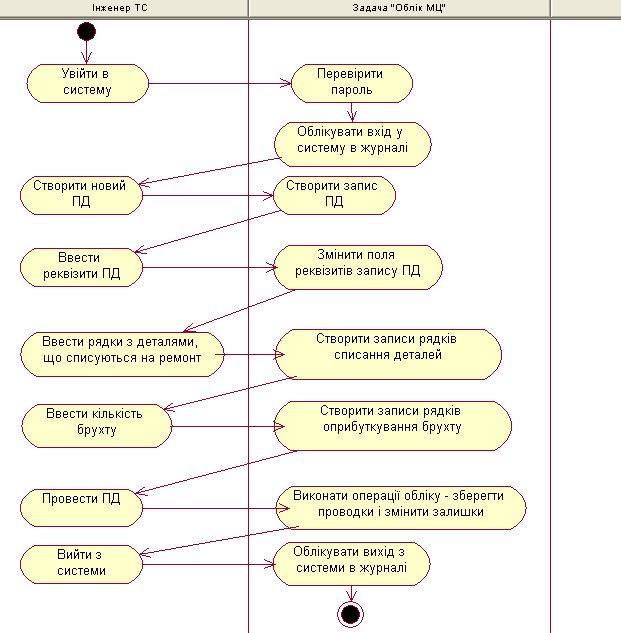


Рисунок 8.3 – Діаграма діяльності для прецеденту *Введення ПД "Дефектна відомість" для ремонту авто*

**Завдання до лабораторної роботи №8**

Для власного варіанту предметної області:

1. Використовуючи потік подій, визначений для певного прецеденту, відображеного на діаграмі послідовності, відобразити процес виконання цього прецеденту на рівні окремих дій у діаграмі діяльності.
2. За результатами виконання роботи оформити звіт.

**Запитання для самоперевірки**

1. Що таке діяльність?
2. Що може робити дія?
3. Як виглядає діаграма Ганта для будівництва?
4. Які типи зв’язків можливі між вузлами?
5. Назвіть інструменти панелі діаграми діяльності та їхнє призначення.
6. Яке співвідношення між поняттями Workflow та Бізнес-процес?
7. Що таке реінжиніринг процесів? У чому його відмінність від реорганізації процесів?
8. Що таке вертикальне стиснення та горизонтальне стиснення процесів?

**Лабораторна робота №9. Створення діаграми компонентів**

**Тема:** Створення діаграми компонентів.

**Мета:** Відобразити ієрархію компонентів програми на діаграмі компонентів.

*Теоретична частина*

**Приклад виконання завдання для предметної області „Технічна служба АТП”.**

Реалізуємо доданок, що автоматизує облік у **Технічній службі АТП**, у середовищі програмування *Ms Access*. Доданок у цьому середовищі працює так:

1. Файл що виконується – *MsAccess.exe*, розташований у папці *Program Files*, один для всіх доданків.
2. Власне специфіка доданку міститься в файлі .mdb, тобто в файлі БД. Тим не менше цей файл містить програмні об’єкти, а саме запити, форми, звіти, веб-сторінки, макроси, модулі. Цей файл фактично є доданком і запускається замість exe-файлу. Насправді при цьому виконується *MsAccess.exe* з параметром – ім’я відповідного файлу доданку.
3. Дані (таблиці з інформацією) беруться з корпоративної мережі, а саме з розташованого в ній сервера БД. Mdb-файл доданку не містить самих таблиць, але містить посилання на таблиці сервера БД.
4. Доданок використовує мінімально необхідну кількість компонентів – усього 3 (див. рис. 9.1).
5. *Visual Basic For Applications* – мова програмування.
6. *Microsoft Access 10.0 Object Library* – візуальні об’єкти середовища програмування, як-от форми, звіти, елементи управління, вікно БД та інше.
7. *Microsoft DAO 3.6 Object Library* – розвинутий інтерфейс доступу до даних, який надає доступ до робочої області (Workspace), БД, таблиць, запитів, наборів даних (recordset), полів, транзакцій і таке інше.

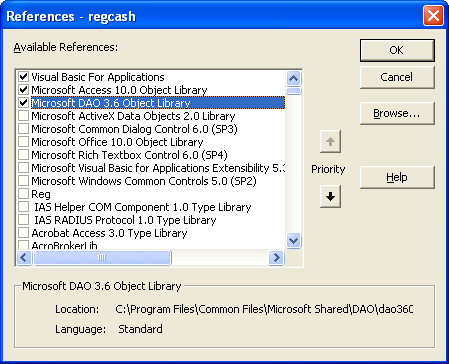


Рисунок 9.1 – Мінімально необхідні компоненти для доданку *MsAccess*

Нехай доданок *MsAccess* буде міститись у файлі *TechService.mdb*, БД на сервері буде мати назву *TechService*. Тоді діаграма компонентів має вигляд (див. рис. 9.2):



Рисунок 9.2 – Діаграма компонентів доданку *TechService* у середовищі *MsAccess*

**Завдання до лабораторної роботи №9**

Для власного варіанту предметної області:

1. Відобразити ієрархію компонентів, які беруть участь у доданку, який може бути розроблений для предметної області. Передбачаємо, що середовище програмування – Visual C++, C# або інше добре знайоме середовище, крім розглянутого у прикладі.
2. Відобразити також системні компоненти доступу до БД, класів інтерфейсу та інші; пояснити їхнє призначення; відобразити також, у яких файлах вони містяться.
3. За результатами виконання роботи оформити звіт.

**Запитання для самоперевірки**

1. Що таке компонент програмного забезпечення?
2. Які переваги дає використання компонентів?
3. Перелічте прикладні і системні компоненти, якими користується Ваш доданок.
4. Що таке динамічне компонування?
5. Чому компоненти повинні поширюватися в двійковій формі?
6. Що таке порти і конектори на діаграмі компонентів?
7. Які існують характеристики „хорошого” компонента?

**Лабораторна робота №10. Створення діаграми розгортання**

**Тема:** Створення діаграми розгортання.

**Мета:** Відобразити діаграму розгортання.

*Теоретична частина*

**Приклад виконання завдання для предметної області „Технічна служба АТП”.**

Як відзначено в описі предметної області, в процесі управління АТП створення окремої БД для технічної служби недоцільне, оскільки перелічена інформація в повному обсязі потрібна для бухгалтерського обліку і знаходитьcя в базі даних інтегрованої АСУ підприємства (ERP-системи). Спробуємо на основі реального проекту АП-2 КиївПасТранс шляхом зворотнього проектування розробити діаграму розгортання ERP-системи управлінського і бухгалтерського обліку автобусного парку.

**Розташування обладнання.**

Таблиця 10.1 – Розташування обладнання, яке використовується в ERP-системі

|  |  |  |  |
| --- | --- | --- | --- |
| *№п.п.* | *Назва приміщення* | *Призначення* | *Обладнання* |
| 1 | Комп’ютерна | Введення дорожніх листів  Звітність по паливу  Нарахування зарплати | 6 роб.станцій:  3 роб.ст.-введення дорожніх листів  2 роб.ст.-нарахування зарплати  1 роб.ст.-звітність по паливу  3 принтери  Сервер |
| 2 | Головний бухгалтер | Облік основних засобів  Підготовка балансу підприємства та іншої звітності | 1роб.ст.  1 принтер |
| 3 | Матеріальна бухгалтерія | Облік палива та інших матеріальних цінностей  Облік талонів на проїзд | 2 роб.ст.:  1 роб.ст.-звітність по обліку матеріалів  1 роб.ст.-облік талонів |
| 4 | Бухгалтерія | Облік грошей в касі  Зв’язок з банком в системі Клієнт-Банк  Облік грошей в банках  Фінансовий облік | 2 роб.ст.  На одній роб.ст. встановлено систему Клієнт-Банк  1 принтер |
| 5 | Склад | Облік матеріалів | 1 роб.ст.  1 принтер |
| 6 | Технічна служба | Облік дефектовочних відомостей, комплектації та розукомплектації транспортних засобів | 1 роб.ст.  1 принтер |

**Комп’ютерна мережа.** Всі комп’ютери з’єднані мережею (10Т-Ethernet) через перемикач (Switch).

**Діаграма розгортання.** Оскільки в діаграмі розгортання міститься досить багато вузлів, для робочих станцій опустимо такі деталі, як встановлені на них компоненти ERP-системи.

На рисунку 9.1 не відображені функції, які виконують робочі станції. Тому необхідним доповненням до нього є таблиця 10.1.



Рисунок 10.1 – Діаграма розгортання системи класу ERP в автобусному парку

**Паралельний порт.** Паралельний порт - тип інтерфейсу, розроблений для комп'ютерів (персональних та інших) для підключення різних периферійних пристроїв. Він також відомий як принтерний порт або порт Centronics (див. рис. 10.2).



Рисунок 10.2 – Паралельний порт DB-25 принтера в стилі IBM-PC і деяких інших комп'ютерів. Цифра 25 означає 25 контактів.

Стереотип зв’язку <<DB-25>> вказується у специфікації зв’язку безпосередньо в полі Stereotype (див. рис. 10.3).

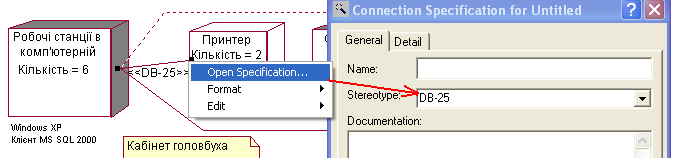


Рисунок 10.3 – Відображення стереотипу для зв’язку між комп’ютером і принтером

**Завдання до лабораторної роботи №10**

Для власного варіанту предметної області:

1. Відобразити уявне розташування пристроїв обробки інформації та обладнання мережі у приміщеннях, в яких буде розгорнута система автоматизації вашої предметної області.
2. За результатами виконання роботи оформити звіт.

**Запитання для самоперевірки**

1. Що таке артефакти? Що ми називаємо артефактами в процесі проектування програмного забезпечення?
2. Які сутності та відповідні діаграми ми розглядаємо на логічному рівні, рівні компонентів, фізичному рівні?
3. Яким чином показати інстальоване програмне забезпечення як процеси?
4. Які типи зв’язків можливі між вузлами?
5. Що таке паралельний порт і для чого він використовується?
6. Як вказати стереотип для зв’язку асоціації між активним та пасивним обладнанням?

# ВИКОРИСТАНА ЛІТЕРАТУРА

1. Роберт Мартін Чиста архітектура. Мистецтво розробки програмного забезпечення, 2019. 368с.
2. Юрій Грицюк Аналіз вимог до програмного забезпечення, 2018. 346с.
3. Ірина Бородкіна, Георгій Бородкин Інженерія програмного забезпечення. Посібник для студентів вищих навчальних закладів, 2018. 230с.
4. Aspen Blake GoF Design Patterns - with examples using Java and UML2, 2022. 249p.
5. Edwin Mach Object Oriented Analysis & Design Cookbook: Introduction to Practical System Modeling, 2019. 218p.
6. Lukman Adam Obomeghie, Bayo Adedeji, Sylvester Aketaumhe Unified Modeling Language Practical Guide, 2022. 59p.
7. Suriya Sundaramoorthy UML Diagramming: A Case Study Approach 1st Edition, 2022. 416p.

# РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Роберт Мартін Чиста архітектура. Мистецтво розробки програмного забезпечення, 2019. 368с.
2. Юрій Грицюк Аналіз вимог до програмного забезпечення, 2018. 346с.
3. Ірина Бородкіна, Георгій Бородкин Інженерія програмного забезпечення. Посібник для студентів вищих навчальних закладів, 2018. 230с.
4. Aspen Blake GoF Design Patterns - with examples using Java and UML2, 2022. 249p.
5. Edwin Mach Object Oriented Analysis & Design Cookbook: Introduction to Practical System Modeling, 2019. 218p.
6. Lukman Adam Obomeghie, Bayo Adedeji, Sylvester Aketaumhe Unified Modeling Language Practical Guide, 2022. 59p.
7. Suriya Sundaramoorthy UML Diagramming: A Case Study Approach 1st Edition, 2022. 416p.

Навчальне видання

(українською мовою)

Лісняк Андрій Олександрович

Мильцев Олександр Михайлович

Мухін Віталій Вікторович

Чопорова Оксана Володимирівна

АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Методичні рекомендації до лабораторних занять

для здобувачів ступеня вищої освіти бакалавра

спеціальності 121 «Інженерія програмного забезпечення»

освітньо-професійної програми «Програмна інженерія»

Рецензент *С. М. Гребенюк*

Відповідальний за випуск *А. О. Лісняк*

Коректор *В. В. Мухін*