3. Проектування програмного забезпечення засобами UML.

3.4 Діаграма послідовності

Діаграма послідовності (sequence diagram) - діаграма, на якій показані взаємодії об'єктів, упорядковані за часом їхнього прояву.

На діаграмі послідовності неявно присутня вісь часу, що дозволяє візуалізувати тимчасові відношення між переданими повідомленнями. За допомогою діаграми послідовності можна представити взаємодію елементів моделі як своєрідний часовий графік "життя" всієї сукупності об'єктів, зв'язаних між собою для реалізації варіанта використання програмної системи, досягнення бізнес-мети або виконання якого-небудь завдання.

На діаграмі послідовності зображуються об'єкти, які безпосередньо беруть участь у взаємодії, при цьому ніякі статичні зв'язки з іншими об'єктами не візуалізуються. Для діаграми послідовності ключовим моментом є саме динаміка взаємодії об'єктів у часі. При цьому діаграма послідовності має як би два виміри. Один – простягається зліва направо у вигляді вертикальних ліній, кожна з яких зображує лінію життя окремого об'єкту, що бере участь у взаємодії. Другий вимір діаграми послідовності - вертикальна тимчасова вісь, спрямована зверху вниз.

Кожен об'єкт графічно зображується у формі прямокутника й розташовується у верхній частині своєї лінії життя (рис. 3.4.1). Усередині прямокутника записуються власне ім'я об'єкта (з малої літери) й ім'я класу, розділені двокрапкою.

Якщо на діаграмі послідовності відсутнє власне ім'я об'єкта, то повинне бути зазначене ім'я класу, до якого він належить. Такий об'єкт вважається **анонімним**. Ім'я класу також може бути відсутнім, але при цьому повинне бути зазначене власне ім'я об'єкта. Такий об'єкт вважається **сиротою**.

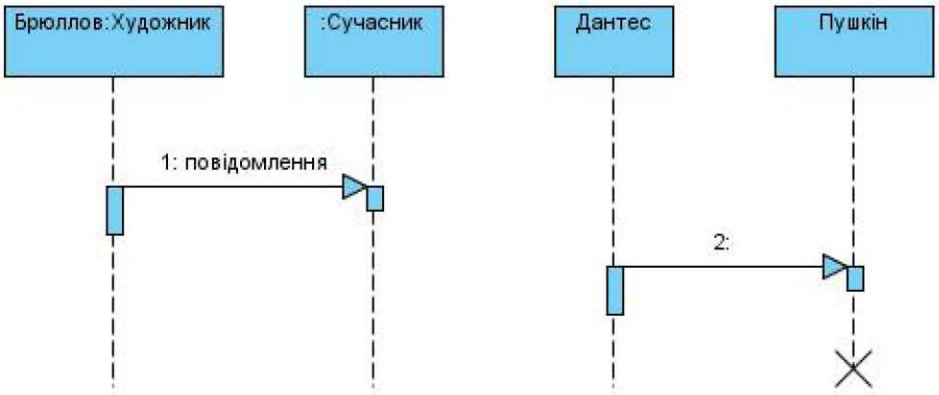

Рис. 3.4.1. Графічні елементи діаграми послідовності

Початковому моменту часу відповідає сама верхня частина діаграми. При цьому процес взаємодії об'єктів реалізується за допомогою повідомлень, що одні об'єкти посилають іншим. Повідомлення зображуються у вигляді горизонтальних стрілок, які можуть мати ім'я повідомлення. Повідомлення утворять певний порядок щодо часу своєї ініціалізації. Інакше кажучи, повідомлення, розташовані на діаграмі послідовності вище, передаються раніше тих, які розташовані нижче. При цьому масштаб на осі часу не вказується, оскільки діаграма послідовності моделює лише часову впорядкованість взаємодій типу "пізніше".

Лінія життя об'єкта (object lifeline) - вертикальна лінія на діаграмі послідовності, що представляє існування об'єкта протягом певного періоду часу.

Лінія життя об'єкта зображується пунктирною вертикальною лінією, асоційованою з єдиним об'єктом на діаграмі послідовності. Лінія життя служить для позначення періоду часу, протягом якого об'єкт існує в системі і може потенційно брати участь у всіх її взаємодіях. Якщо об'єкт існує в системі постійно, то і його лінії життя триває по всій робочій області діаграми послідовності від самої верхньої її частини до самої нижньої (об'єкт Користувач Петро на рис. 3.4.2).

Окремі об'єкти, закінчивши виконання своїх операцій, можуть бути знищені , щоб звільнити займані ними ресурси. Для таких об'єктів лінія життя обривається в момент їх знищення. Для позначення моменту знищення об'єкта застосовується спеціальний символ у формі латинської букви "X". На рис 3.4.2 цей символ використовується для знищення об'єкта «Повідомлення про помилку», утвореного від об'єкту «Клієнтська сессія» . Нижче цього символу пунктирна лінія не зображується, оскільки відповідного об'єкта в системі вже немає, і цей об'єкт повинен бути виключений із всіх наступних взаємодій

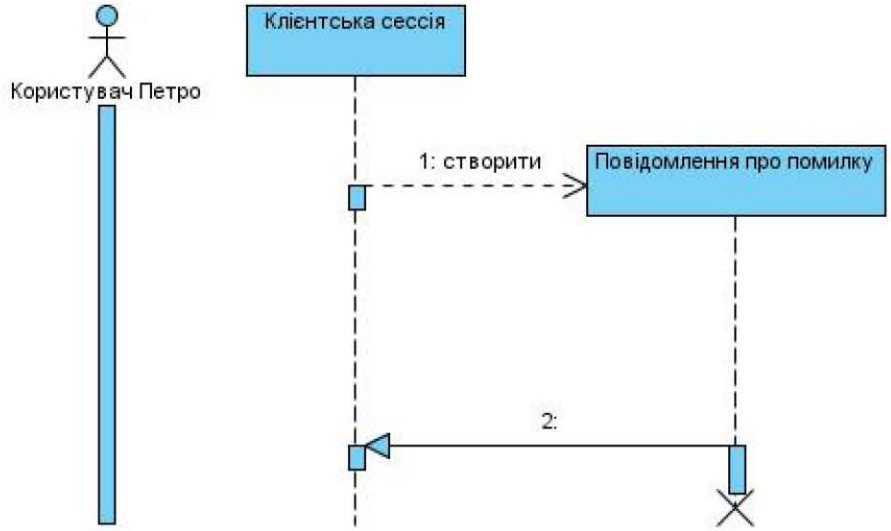

Рис. 3.4.2. Графічне зображення ліній життя й фокусів керування об'єктів

Іноді можна зустріти діаграму послідовностей яка містить лінію життя, що починається елементом Актор. Таким чином підкреслюється зв'язок із варіантом використання.
Аналогічно можна зустріти позначення граничного, керуючого та класу-сутності.

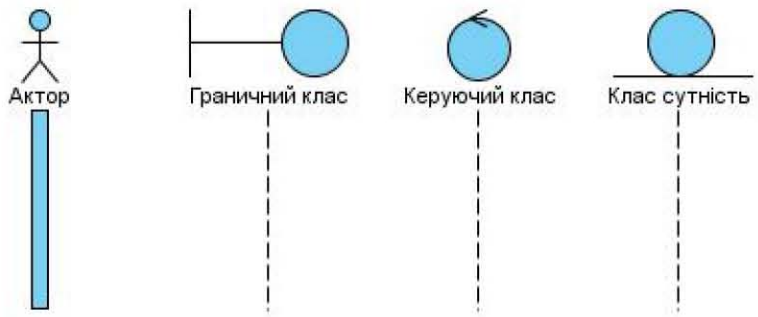

Рис. 3.4.3. Графічне зображення актора, граничного класу, керуючого та класа-сутності на діаграмі послідовностей

Зовсім не обов'язково створювати всі об'єкти в початковий момент часу. Окремі об'єкти в системі можуть створюватися в міру необхідності, істотно заощаджуючи ресурси системи й підвищуючи її продуктивність. У цьому випадку прямокутник такого об'єкта зображується не у верхній частині діаграми послідовності, а в тій, котра відповідає моменту створення об'єкта (об'єкт «Повідомлення про помилку», утворений від об'єкту «Клієнтська сессія»). Об'єкт створюється зі своєю лінією життя а, можливо, і з фокусом керування.

У процесі функціонування об'єктно-орієнтованих систем одні об'єкти можуть перебувати в активному стані, безпосередньо виконуючи певні дії, або в стані пасивного очікування повідомлень від інших об'єктів.

Фокус керування (focus of control) - спеціальний символ на діаграмі послідовності, що вказує період часу, протягом якого об'єкт виконує деяку дію, перебуваючи в активному стані.

Фокус керування зображується у формі витягнутого вузького прямокутника (об'єкт Клієнтська сессія на рис 3.4.2), верхня сторона якого позначає початок одержання фокуса управління об'єкта (початок активності), а її нижня сторона - закінчення фокуса керування (закінчення активності). Цей прямокутник розташовується нижче позначення відповідного об'єкта й може заміняти його лінію життя (об'єкт Користувач Петро на рис. 3.4.2), якщо на всьому її протязі він активний.

Періоди активності об'єкта можуть чергуватися з періодами його пасивності або очікування. У цьому випадку в такого об'єкта фокуси керування змінюють своє зображення на лінію життя й навпаки (об'єкт Клієнтська сессія на рис 3.4.2). Важливо розуміти, що одержати фокус керування може тільки об'єкт, у якого в цей момент є лінія життя. Якщо ж об'єкт був знищений, то знову виникнути в системі він уже не може. Замість нього може бути створений лише екземпляр цього ж класу, що, строго кажучи, буде іншим об'єктом.

В окремих випадках ініціатором взаємодії в системі може бути актор або зовнішній користувач. При цьому актор зображується на діаграмі послідовності найпершим об'єктом ліворуч зі своїм фокусом керування (рис. 3.4.2). Найбільше часто актор і його фокус керування будуть існувати в системі постійно, відзначаючи характерну для користувача активність в ініціюванні взаємодій із системою. Актор може мати власне ім'я або залишатися анонімним.

В окремих випадках об'єкт може посилати повідомлення самому собі, ініціюючи так називані **рефлексивні** повідомлення. Для цієї мети призначене спеціальне зображення (повідомлення об'єкта а на рис 3.4.4). Такі повідомлення зображуються у формі повідомлення, початок і кінець якого стикаються з лінією життя або фокусом керування того самого об'єкта. Подібні ситуації виникають, наприклад, при обробці натискань на клавіші клавіатури при уведенні тексту в документ, що редагується, при наборі цифр номеру телефону абонента.

Якщо в результаті рефлексивного повідомлення створюється новий підпроцес або гілка керування, то говорять про рекурсивний або вкладений фокус керування. На діаграмі послідовності рекурсія позначається невеликим прямокутником, приєднаним до правої сторони фокуса керування того об'єкта, для якого зображується дана рекурсивна взаємодія (об'єкт б на рис 3.4.4).

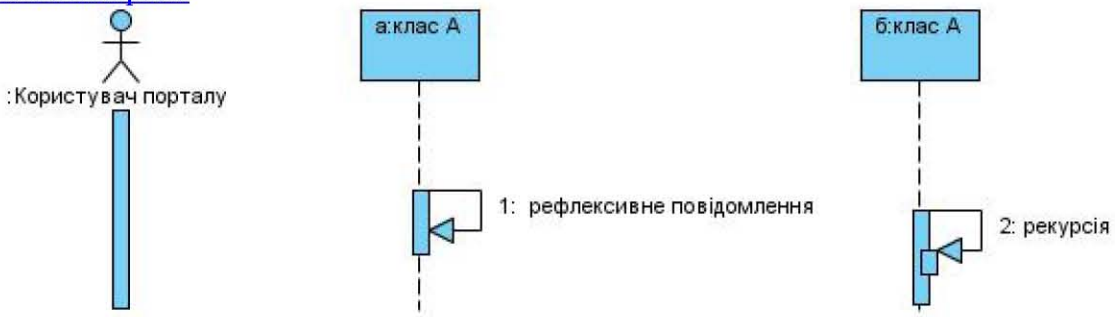

Рис. 3.4.4. Графічне зображення актора, рефлексивного повідомлення й рекурсії на діаграмі послідовності

Повідомлення – сигнал до виконання певної операція, яка повинна бути здійснена об'єктом, що прийняв повідомлення.

На діаграмах послідовності можуть бути присутнім три різновиди повідомлень, кожне з яких має своє графічне зображення (рис 3.4.5).

Рис. 3.4.5. Графічне зображення різних видів повідомлень між об'єктами на діаграмі послідовності

Перший різновид повідомлення (рис 3.4.5, 2:виклик процедури) найпоширеніший. Він використовується для виклику процедур, виконання операцій або позначення окремих вкладених потоків керування. Початок цієї стрілки, як правило, стикається з фокусом керування того об'єкта-клієнта, що ініціює це повідомлення. Кінець стрілки стикається з лінією життя того об'єкта, що приймає це повідомлення й виконує у відповідь певні дії. При цьому приймаючий об'єкт може одержати фокус керування, стаючи в цьому випадку активним. Передаючий об'єкт може втратити фокус керування або залишитися активним.

Другий різновид повідомлення (рис 3.4.5, 1:асинхронний виклик) використовується для позначення простого асинхронного повідомлення, що передається в довільний момент часу.

Третій різновид повідомлення (рис 3.4.5, 3:повернення з процедури) використовується для повернення з виклику процедури. Прикладом може служити просте повідомлення про завершення обчислень без надання результату розрахунків об'єкту-клієнтові. У процедурних потоках керування ця стрілка може бути опущена, оскільки її наявність неявно передбачається наприкінці активізації об'єкта. У той же час вважається, що кожен виклик процедури має свою пару - повернення виклику. Для непроцедурних потоків керування, включаючи паралельні й асинхронні повідомлення, стрілка повернення повинна вказуватися явно.

Стереотипи повідомлень

У мові UML передбачені деякі стандартні дії, що виконуються у відповідь на отримання відповідного повідомлення. Вони можуть бути явно вказані на діаграмі послідовності у формі стереотипу поряд з повідомленням, до якого вони відносяться. В цьому випадку вони записуються в лапках. Використовуються наступні позначення для моделювання дій:

- "call" (викликати) - повідомлення, що вимагає виклику операції або процедури приймаючого об'єкту. Якщо повідомлення з цим стереотипом рефлексивне, то воно ініціює локальний виклик операції у того самого об’єкта, що надіслав повідомлення;

- "return" (повернути) - повідомлення, що повертає значення виконаної операції або процедури об'єкту, що викликав її. Значення результату може ініціювати розгалуження потоку управління;

- "create" (створити) - повідомлення, що вимагає створення іншого об'єкту для виконання певних дій. Створений об'єкт може отримати фокус управління, а може і не отримати його;

- "destroy" (знищити) - повідомлення з явною вимогою знищити відповідний об'єкт. Посилається у тому випадку, коли необхідно припинити небажані дії з боку об'єкту, що існує в системі, або коли об'єкт більше не потрібний і повинен звільнити задіяні їм системні ресурси;

- "send" (послати) - позначає посилку іншому об'єкту деякого сигналу, який асихронно ініціюється одним об'єктом і приймається (перехоплюється) іншим. Відмінність сигналу від повідомлення полягає в тому, що сигнал повинен бути явно описаний в тому класі, об'єкт якого ініціює його передачу.

Нижче представлена діаграма послідовності із використанням стереотипних значень (рис. 3.4.6). На діаграмі проілюстровано скорочену варіацію на тему казки про «Ріпку». Зауважимо, що на початку ріпки в системі не існувало, даний об’єкт був посаджений (створений) дідом, що відображається стереотипом «create». Після невдалою спробі витягти ріпку, дід звернувся до бабці (стереотип «send»), яка в свою чергу покликала мишу(стереотип «send»). Знищення ріпки відображено стереотипом «destroy».

Окрім стереотипів, повідомлення можуть мати власне позначення операції, виклик якої вони ініціюють у приймаючого об'єкту. В цьому випадку поряд із стрілкою записується ім'я операції з круглими дужками, в яких можуть указуватися параметри або аргументи відповідної операції. Якщо параметри відсутні, то дужки все одно повинні бути присутніми після імені операції. Прикладами таких операцій можуть служити наступні: "видати клієнтові готівкою суму (n)", "встановити з'єднання між абонентами (a, b)", "зробити текст, що вводиться, невидимим ()", "подати звуковий сигнал тривоги ()".

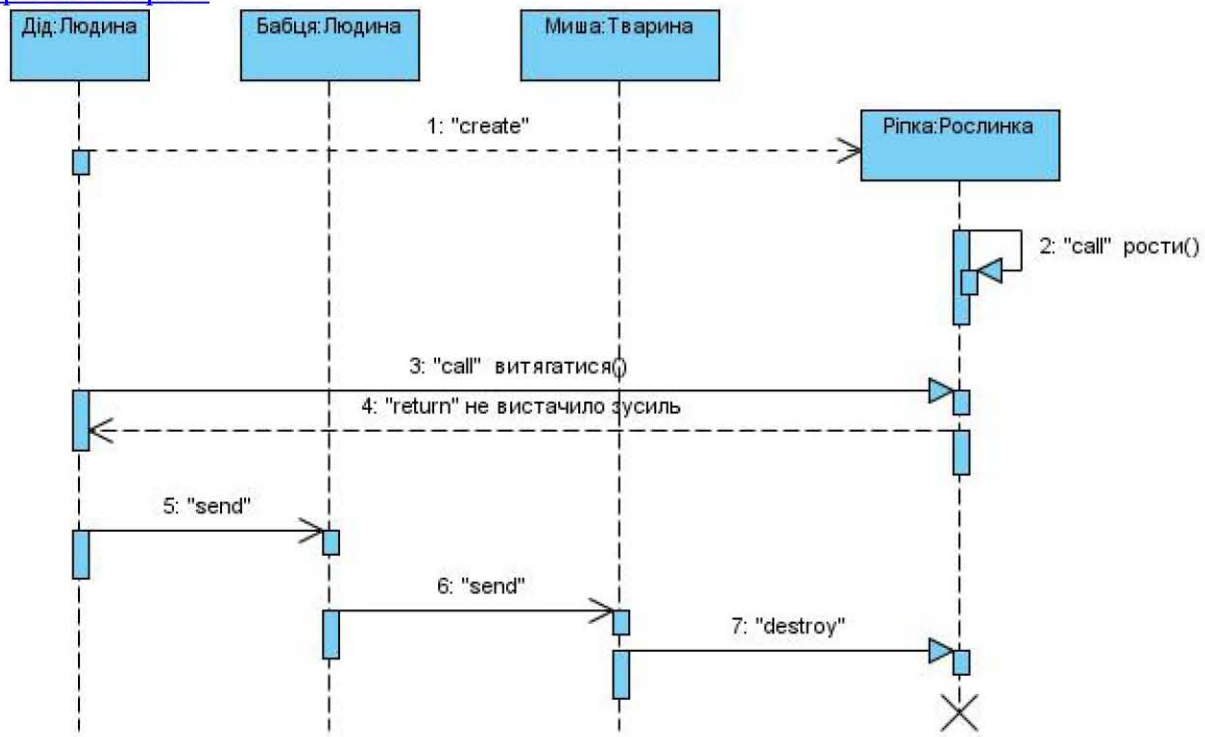

Рис. 3.4.6. Діаграма послідовностей із стереотипними значеннями повідомлень

Загублені та знайдені повідомлення

Загублені повідомлення(Lost messages) – це повідомлення, які надсилаються, проте не надходять до зазначеного отримувача або отримувач не наявний на даній діаграмі. Знайдені повідомлення(Found messages) – це повідомлення, що надходять від невідомого відправника або відправник не наявний на даній діаграмі. На наступній діаграмі представлено їх графічне зображення.

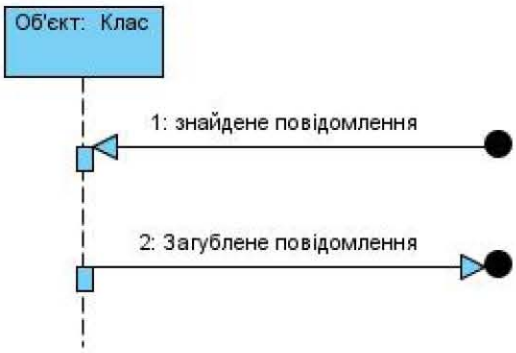

Рис. 3.4.7. Діаграма послідовностей із загубленими та знайденими повідомленнями

Розгалуження потоку керування

Одна з особливостей діаграми послідовності - можливість візуалізувати розгалуження процесу. Для зображення розгалуження використовуються дві або більше стрілки, що виходять із однієї крапки фокуса управління об'єкта (об'єкт 1 на рис. 3.4.8). Поруч із кожною з них у формі булевського виразу повинна бути явно зазначена відповідна умова гілки.

Кількість гілок може бути довільною, однак наявність розгалужень може істотно ускладнити інтерпретацію діаграми послідовності. Речення-умова повинне бути явно зазначене для кожної гілки, воно записується у формі звичайного тексту, псевдокоду або виразу на мові програмування. Цей вираз завжди має повертати деякий булевський вираз. Запис цих умов повинен виключати одночасну передачу альтернативних повідомлень по двом і більше гілкам. У противному випадку на діаграмі послідовності може виникнути конфлікт розгалуження.

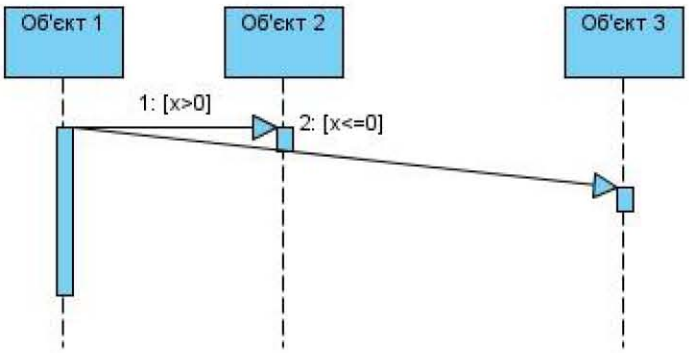

Рис. 3.4.8. Графічне зображення бінарного розгалуження потоку керування на діаграмі послідовності

Діаграми послідовностей не розраховані на те, щоби ілюструвати складну процедурну логіку. Згідно до стандарту UML2.0 в цьому випадку використовують механізми, що дозволяють скомбінувати діаграму з додатковими фрагментами. Ці фрагменти також є послідовностями, які розташовуються у спеціалізованому фреймі та виконуються за певних умов. Можливі наступні фрагменти:

- Альтернативні фрагменти (позначаються “alt”) моделюють if…then…else конструкції.
- Опційні (позначаються “opt”) моделюють switch конструкції.
- Розриваючі (Break) фрагменти моделюють альтернативну послідовність подій до тієї, що зображена на базовій діаграмі.
- Паралельні фрагменти (позначаються “par”) моделюють паралельні процеси.
- Слабкі послідовні фрагменти (позначаються “seq”) містять послідовності, в яких всі повідомлення мають бути оброблені до того, як розпочнеться наступний сегмент, проте всередині даного фрагменту порядок обробки повідомлень, що не пов’язані лінією життя, може бути довільним.
- Сильні послідовні фрагменти (позначаються “strict”) містять групу повідомлень, що мають бути оброблені в заданому порядку.
- Заперечуючі фрагменти (позначаються “neg”) містять неприйнятні послідовності повідомлень.
- Критичні фрагменти містять критичні ділянки процесів.
- Ігноруючі фрагменти оголошують, що повідомлення не має значення, якщо з’являється в даному контексті.
- Прийнятні фрагменти мають зміст протилежний до ігноруючих – будь-яке повідомлення, що не включене до прийнятного фрагменту ігнорується.
- Контролюючі фрагменти (позначаються “assert”) зазначають, що будь-яка послідовність, що не показана як операнд операції контролю істинності є непрацездатною.
- Циклічні фрагменти містять послідовність повідомлень, що повторюється.

Нижче наведено діаграми, що ілюструють альтернативні фрагменти (рис. 3.4.9) та циклічні (рис 3.4.10).

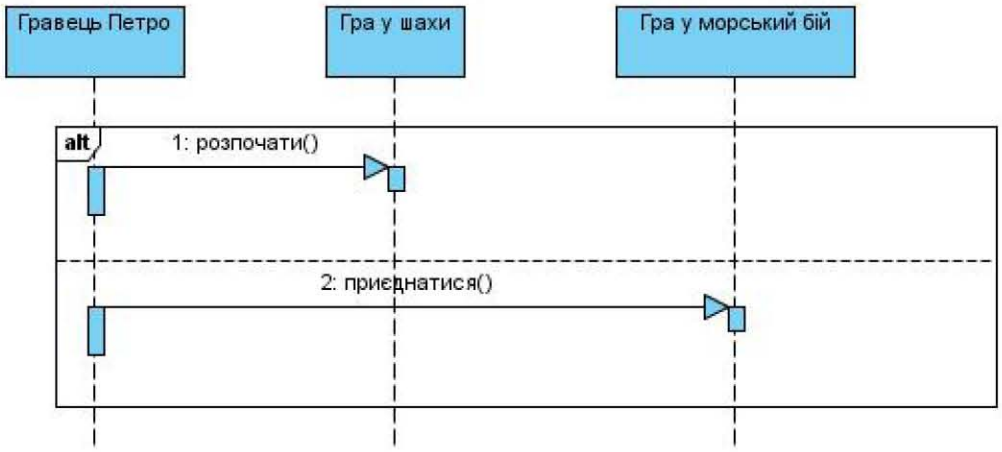

Рис. 3.4.9. Графічне зображення альтернативного фрагменту на діаграмі послідовності

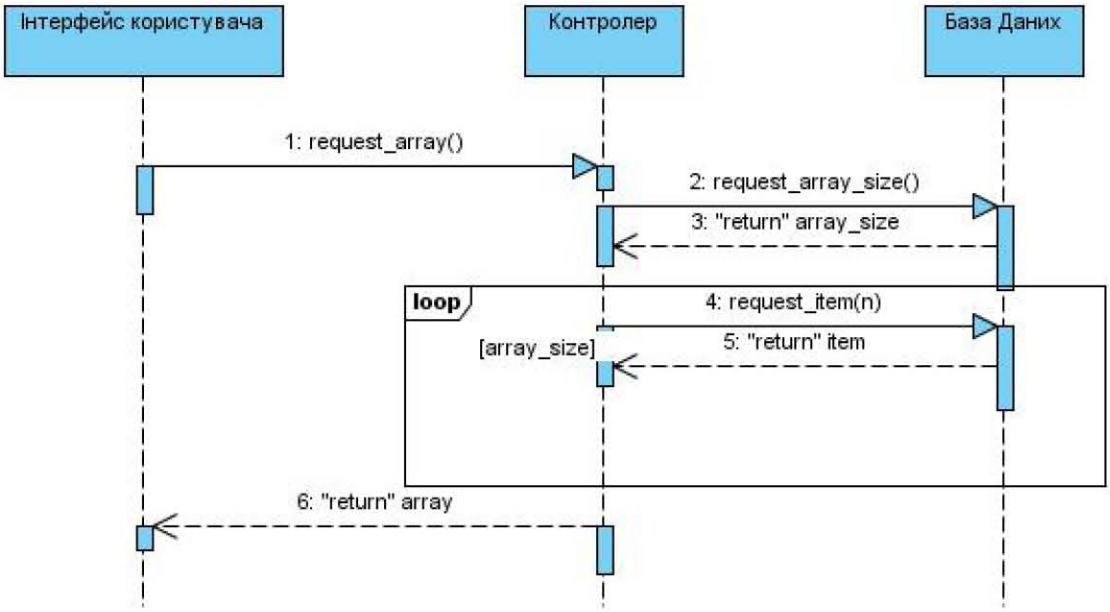

Рис. 3.4.10. Графічне зображення циклічного фрагменту на діаграмі послідовності

Часткова декомпозиція

Згідно зі стандартом, об'єкт може мати більше однієї лінії життя, що виходять з нього (рис 3.4.11 Об'єкт :Клас). Це дозволяє зображати на діаграмі внутрішні повідомлення. Проте, зазначимо, що насьогодні доволі мала кількість середовищ розробки підтримує такий функціонал.

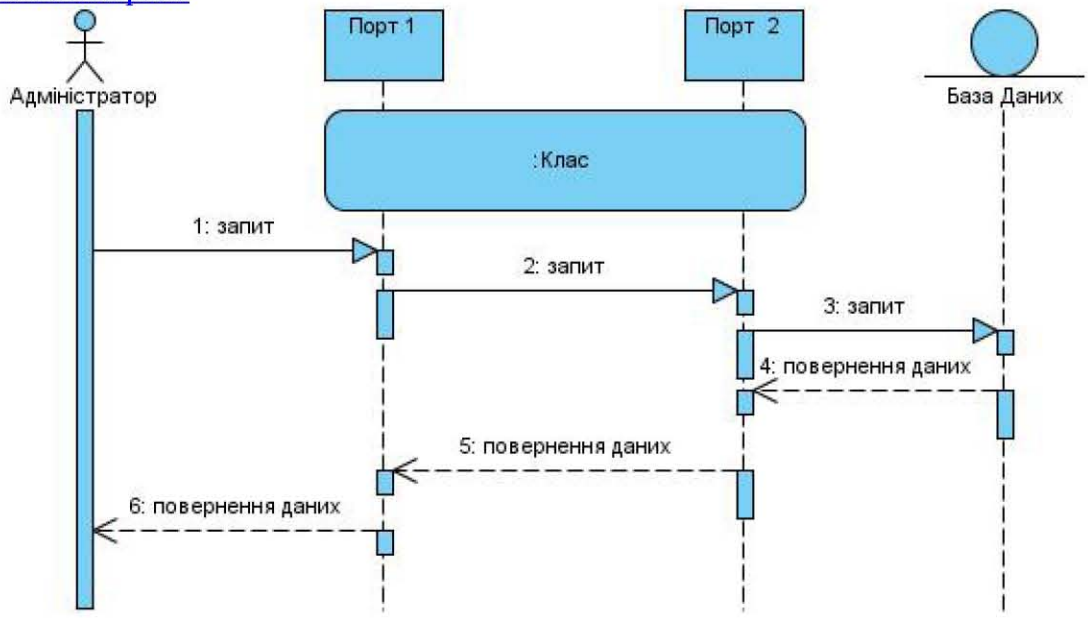

Рис. 3.4.11. Графічне зображення часткової декомпозиції на діаграмі послідовності

Приклад:

Для наочності матеріалу, розглянемо діаграму послідовностей, яка буде відповідати діаграмі класів, розглянутій вище (рис 3.3.22) та ілюструвати казку «Курка Ряба» (рис 3.4.12).

Курка Ряба знесла (стереотип «create») Золоте Яйце. Дідусь та Бабця намагалися розбити яйце (циклічний фрагмент), проте їхні спроби не були вдалими. Бігло Мишеня (рекурсивне повідомлення бігти()), махало хвостиком (рекурсивне повідомлення махати хвостиком()) та розбило яйце (стереотип «destroy»). Дідусь та Бабця плакали, а Курка Ряба знесла нове Яйцо, Просте (стереотип «create»).

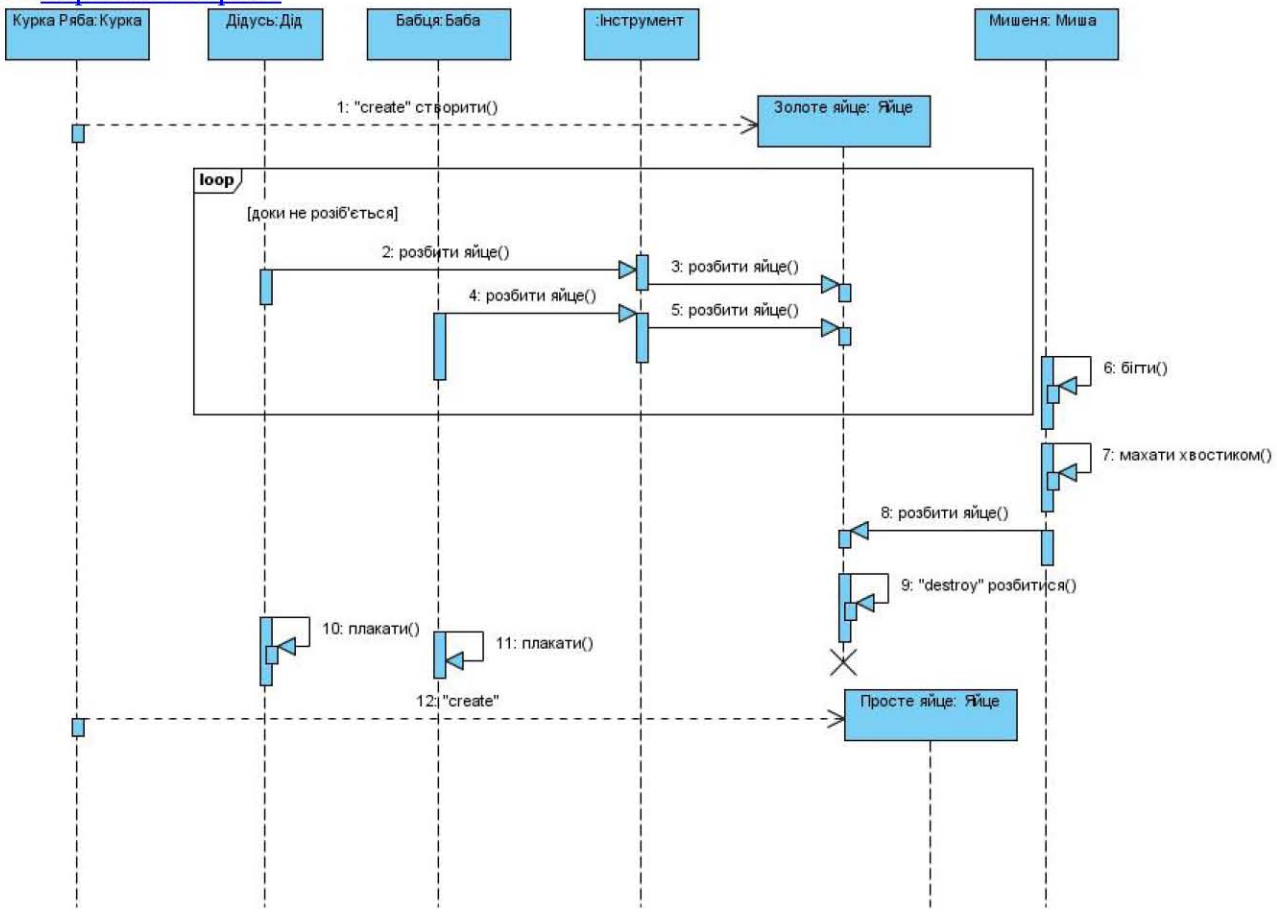

Рис. 3.4.12. Діаграма послідовностей для ілюстрації казки «Курка Ряба»

За допомогою UML можливо отримати формальний опис проекту, створити його архітектуру, проте суто UML не дає відповіді, наскільки структура якісна, чи вдало використаний об'єктно-орієнтований підхід. Розглянемо ці питання детальніше