ЛАБОРАТОРНА РОБОТА №2. ГРАФІЧНА НОТАЦІЯ UML, ДОКУМЕНТУВАННЯ ПРОЕКТУ

Тема: Документування проекту.

Мета: Ознайомлення з видами діаграм UML. Отримання базових навичок з використання діаграми класів мови UML. Здобуття навичок з використання засобів автоматизації UML-моделювання на прикладі ArgoUML/Umbrello. Документування проекту за допомогою JavaDoc.

Довідка

UML (англ. Unified Modeling Language) – уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML призначена для візуалізації, проектування, моделювання та документування при розробці програмного забезпечення. UML не є мовою програмування, але існують засоби кодогенерації на основі UML. UML складається з 13 діаграм, що поділяються на структурні та поведінкові.

Структурні діаграми:

- Класів – Class diagram;
- Компонент – Component diagram;
- Композитної/складеної структури – Composite structure diagram (Кооперації – Collaboration diagram (UML2.0));
- Розгортування – Deployment diagram;
- Об'єктів – Object diagram;
- Пакетів – Package diagram;

Діаграми поведінки:

- Діяльності – Activity diagram;
- Скінчених автоматів (станів) – State Machine diagram;
- Прецедентів – Use case diagram;
- Діаграми взаємодії:
  - Кооперації – Collaboration (UML 1.x) / Комунікації – Communication (UML2.0);
  - Огляду взаємодії – Interaction overview diagram (UML2.0);
  - Послідовності – Sequence diagram;
  - Синхронізації – UML Timing Diagram (UML2.0).

Діаграма класів – статичне представлення структури моделі. Подає статичні (декларативні) елементи, такі як: класи, типи даних, їх зміст та відношення. Діаграма класів, також, може містити позначення для пакетів та може містити позначення для вкладених пакетів. Також, діаграма класів може містити позначення деяких елементів поведінки, однак, їх динаміку розкрито в діаграмах інших типів.

Клас позначається прямокутником з трьох частин: верхня містить ім’я класу, середня список атрибутів класу, нижня – список операцій класу. Додатково елементи списків можуть позначатись типами та областю видимості.

Відношення між класами позначається різними видами ліній і стрілок. Існують такі види відношень:

- Асоціація – показує, що об'єкти однієї сутності (класу) пов'язані з об'єктами іншої сутності. Позначається суцільною лінією. Може бути унарою – односторонньою (позначається відкритою стрілкою, що вказує на напрям асоціації) та бінарною. В асоціації можуть брати участь більше двох класів, такі асоціації позначаються лініями, один кінець яких йде до класового блоку, а інший до загального ромбу. Асоціації можуть бути іменованими, тоді на кінцях її лінії позначені ролі, приналежності, індикатори, мультиплікатори, видимості або інші властивості.
- Агрегація – різновид асоціації, при відношенні між цілим і його частинами. Як тип асоціації, агрегація може бути іменованою. Агрегація не може включати відразу декілька класів. Агрегація зустрічається, коли один клас є колекцією або контейнером інших. Час існування класів-частин не залежить від часу існування класу-цілого. Якщо контейнер буде знищений, то його вміст – ні. Графічно агрегація позначається порожнім ромбом на блоці класу-цілого і лінією, що йде від цього ромба до класу-частини.
- Композиція – більш суворий варіант агрегації. Під час композиції має місце жорстка залежність часу існування класу-цілого та класів-часток. Якщо контейнер буде знищений, то весь його вміст буде також знищено. Графічно позначається як і агрегація, але з зафарбовані ромбом.
- Узагальнення (Наслідування) – показує, що один з двох зв'язаних класів (підтип) є більш конкретною формою іншого (супертипу), який називається узагальненням першого. На практиці це означає що будь-який екземпляр підтипу є також примірником супертипу. Графічно генералізація представляється лінією з закритою стрілкою (порожнім трикутником) у супертипа. Генералізація також відома як наслідування або зв'язок типу "is a".
- Реалізація – відношення між двома елементами моделі, в якому один елемент (клієнт) реалізує поведінку, задану іншим (постачальником). Графічно реалізація представляється також як і генералізація, але з пунктирною лінією.
- Залежність – це відношення використання, при якому зміна в специфікації одного тягне за собою зміну іншого, причому протилежне не обов'язково. Графічно позначається пунктирною стрілкою, що йде від залежного елементу до того, від якого він залежить.

Javadoc – генератор документації в HTML-форматі з коментарів сирцевого коду на Java від Sun Microsystems. Javadoc – стандарт для документування класів Java. Javadoc також надає API для створення доклетів і теглетів, які дозволяють програмісту аналізувати структуру Java- додатку.

Коментарі документації застосовують для документування класів, інтерфейсів, полів (змінних), конструкторів, методів та пакетів. У кожному разі коментар повинен знаходитися перед елементом, що документується. Дескриптори Javadoc починаються з символу «@».

Завдання

1. Ознайомитись з призначенням та видами діаграм мови UML. Вивчити діаграму класів, вільно володіти елементами та відношеннями між ними. Вміти будувати діаграми класів для сирцевого коду Java, а також генерувати програмний код еквівалентний заданій діаграмі класів.
2. Побудувати діаграму класів, що містить три інтерфейси Ifl, If2, If3 з методами methl(), meth2(), meth3 та класи що їх реалізують Cl1, Cl2, Cl3 відповідно.
3. Згідно варіанту (нижче) реалізувати на діаграмі класів відношення генералізації та агрегації.
4. В підготованому проекті (ЛР1) створити програмний пакет com.lablll.labwork2. В пакеті розробити інтерфейси і класи згідно діаграмі (п.3-4). Реалізація методів має виводити на консоль ім'я класу та назву методу).
5. Ознайомитись із засобами автоматизації UML-моделювання. Вміти використовувати середовища ArgoUML та Umbrello на базовому рівні для розробки діаграми класів та документування програмного забезпечення.
6. За допомогою середовища ArgoUML або Umbrello імпортувати сирцеві коди пакету com.lablll.labwork2 та перевірити відповідність побудованої діаграми класів з розробленою (п.3-4). Зберегти діаграму в каталозі документації проекту.
7. Ознайомитись з синтаксисом коментарів для засобу автоматизації документації JavaDoc. Модифікувати сирцеві коди пакету com.labl 11 .labwork2 додавши коментарі у форматі JavaDoc.
8. Згенерувати JavaDoc за допомогою Eclipse (меню Project) у каталог документації проекту.
9. Розробити ціль ANT для генерації JavaDoc. Згенерувати JavaDoc за допомогою розробленої цілі ANT.

Варіанти завдання

Генералізація (наслідування)

Номер варіанту завдання обчислюється як залишок від ділення номеру залікової книжки на 9.

0. If1 <- If2; If2 <- If3; Cl1 <- Cl3
1. If1 <- If3; If3 <- If2; Cl1 <- Cl2
2. If1 <- If2; If1 <- If3; Cl2 <- Cl3
3. If2 <- If1; If1 <- If3; Cl1 <- Cl3
4. If2 <- If3; If3 <- If1; Cl2 <- Cl1
5. If2 <- If1; If2 <- If3; Cl2 <- Cl3
6. If3 <- If2; If2 <- If1; Cl2 <- Cl1
7. If3 <- If1; If1 <- If2; Cl3 <- Cl1
8. If3 <- If2; If3 <- If1; Cl3 <- Cl2

Агрегація

Номер варіанту завдання обчислюється як залишок від ділення номеру залікової книжки на 5.

0. I1 <- Cl1; Cl3 <- Cl2; Cl3 <- Cl3
1. I1 <- Cl2; Cl1 <- Cl3; Cl1 <- Cl1
2. I1 <- Cl3; Cl1 <- Cl2; Cl1 <- Cl1
3. I1 <- Cl1; I2 <- Cl1; Cl3 <- Cl2
4. I3 <- Cl2; I2 <- Cl3; Cl3 <- Cl1

Протокол

Протокол має містити титульну сторінку, завдання, роздруківку діаграми класів, розроблений програмний код та згенеровану документацію в форматі JavaDoc.