1. Właściwość statyczna

Atrybut/metoda wspólna dla wszystkich obiektów danej klasy, zamiast być przypisana do konkretnej instancji.

Oznacza to, że:

* **Nie jest powiązana z instancją** – można jej używać bez tworzenia obiektu danej klasy.
* **Jest przechowywana w jednej kopii** – niezależnie od liczby utworzonych obiektów danej klasy, istnieje tylko jedna wersja tej właściwości.
* **Jest częścią klasy, a nie instancji** – oznacza to, że dostęp do niej uzyskujemy przez nazwę klasy.

**Kiedy stosujemy właściwości statyczne?**

* **Gdy dana wartość powinna być wspólna dla wszystkich obiektów**
* **Gdy dana metoda nie operuje na konkretnej instancji, ale na ogólnej logice klasy**
* **Optymalizacja pamięci** – statyczne pola mogą zmniejszyć zużycie pamięci, jeśli nie ma potrzeby replikowania wartości dla każdej instancji.

**Konsekwencje stosowania statycznych właściwości**

* **Brak dostępu do this** – metoda statyczna nie ma dostępu do pól i metod instancyjnych, ponieważ nie jest powiązana z konkretną instancją.

W programie Visual Paradigm właściwość statyczną tworzymy poprzez ustawienie pola Scope w Atrtibute Specification na wartość classifier zamiast instance.

W programie Visual Paradigm właściwość statyczną jest oznaczana poprzez podkreślenie.

1. Właściwość niemodyfikowana (read only)

To atrybut, którego wartość można ustawić tylko raz – podczas inicjalizacji. Oznacza to, że po przypisaniu wartości nie można jej już zmienić w trakcie działania programu.

Oznacza to, że:

* **Jest przypisana tylko raz** – wartość właściwości niemodyfikowanej można ustawić tylko podczas deklaracji lub w konstruktorze i nie można jej zmieniać później.
* **Nie można jej nadpisać** – każda próba modyfikacji po inicjalizacji spowoduje błąd kompilacji.
* **Zachowuje spójność danych** – wartości są chronione przed przypadkowymi zmianami, co zapewnia większą stabilność kodu.

**Kiedy stosujemy właściwości niemodyfikowane?**

* **Gdy wartość powinna być stała po utworzeniu obiektu** – np. identyfikator użytkownika (ID), data utworzenia obiektu.
* **Ochrona przed przypadkowymi zmianami** – zapobiega nadpisywaniu kluczowych informacji w kodzie.
* **Lepsza czytelność kodu** – jeśli atrybut jest oznaczony jako read-only, programista od razu wie, że jego wartość się nie zmienia.

**Konsekwencje stosowania właściwości niemodyfikowanych**

* **Brak możliwości modyfikacji po inicjalizacji**
* **Trzeba przemyśleć inicjalizację** – wartość musi być przypisana od razu w konstruktorze lub podczas deklaracji

W programie Visual Paradigm właściwość niemodyfikowalną tworzymy poprzez zaznaczenie okienka Read only w Attribute Specification.

W programie Visual Paradigm właściwość niemodyfikowalna jest oznaczana poprzez dodanie {readOnly} po nazwie, liczności, typie i ewentualnej wartości atrybutu

1. Właściwość wnioskowana (derived property)

**Właściwość wnioskowana (derived property)** to atrybut, którego wartość jest obliczana na podstawie innych właściwości klasy, zamiast być przechowywana bezpośrednio w pamięci. Nie ma fizycznego odpowiednika w strukturze danych obiektu, ale jej wartość jest dynamicznie zwracana przez metodę lub wyrażenie.

Oznacza to, że:

* **Nie jest przechowywana bezpośrednio** – jej wartość nie istnieje jako oddzielne pole w pamięci, lecz jest obliczana na żądanie.
* **Jest dynamicznie wyliczana** – jej wartość zawsze wynika z aktualnych danych obiektu.
* **Nie można jej jawnie przypisać** – wartość pochodzi z obliczeń, więc nie można jej zmieniać bezpośrednio.
* **Zawsze odzwierciedla aktualny stan obiektu** – jeśli zmienią się wartości, na podstawie których jest obliczana, jej wynik automatycznie się zmieni.

**Kiedy stosujemy właściwości wnioskowane?**

* **Gdy wartość można wyliczyć na podstawie innych danych** – np. pełna nazwa (fullName) na podstawie firstName i lastName.
* **Gdy nie chcemy przechowywać redundantnych informacji** – zamiast przechowywać długość listy, można ją obliczać dynamicznie.
* **Gdy chcemy, aby wartość zawsze była aktualna** – zmiany w powiązanych właściwościach automatycznie wpływają na wartość wnioskowaną.
* **Konsekwencje stosowania właściwości wnioskowanych**
* **Nie zajmuje dodatkowej pamięci**
* **Zawsze aktualna**
* **Może wymagać większej liczby operacji obliczeniowych**

W programie Visual Paradigm właściwość wnioskowaną tworzymy poprzez zaznaczenie okienka Derived w Attribute Specification.

Możemy też dodać formułę obliczająca dany atrybut w zakładce Tagged Values, (również w Attribute Specification) dodając tam nową formułę.

Visual Paradigm oznacza właściwości wnioskowane za pomocą / przed nazwą atrybutu

1. Wartość domyślna właściwości

**Wartość domyślna właściwości** to początkowa wartość przypisana do atrybutu klasy w momencie jego tworzenia. Jeśli użytkownik nie poda innej wartości, zostanie użyta wartość domyślna.

### **Oznacza to, że:**

* **Jest ustawiana automatycznie** – jeśli nie podamy innej wartości, zostanie użyta domyślna.
* **Może być nadpisana** – użytkownik może zmienić wartość później, chyba że atrybut jest final  lub readonly .
* **Zwiększa przewidywalność kodu** – zapewnia spójność i minimalizuje błędy wynikające z niezainicjalizowanych wartości.

**Kiedy stosujemy wartość domyślną?**

* **Aby uniknąć wartości null lub niezainicjalizowanych zmiennych** – zapewnia to stabilność programu.
* **Gdy większość obiektów powinna mieć określoną wartość** – np. isActive = true w systemach rejestracji użytkowników.
* **Aby uprościć konstruktor** – użytkownik nie musi przekazywać wszystkich wartości podczas tworzenia obiektu.

### **Konsekwencje stosowania wartości domyślnej**

* **Zmniejsza liczbę błędów** – zapobiega przypadkowemu użyciu niezainicjalizowanych wartości.
* **Poprawia wygodę programowania** – użytkownik nie musi zawsze podawać wszystkich parametrów.
* **Może wymagać ostrożności** – jeśli wartość domyślna jest nieodpowiednia, może prowadzić do błędów logicznych w programie.

W programie Visual Paradigm wartość domyślną właściwości możemy ustawić poprzez podanie tej wartości w polu Initial Value w Attribute Specification.

Visual Paradigm pokazuje wartość domyślną wwłaściwości na diagramie za nazwą i typem danego atrybutu i po znaku „=”

1. Krotność i liczność

**Krotność** określa minimalną i maksymalną liczbę obiektów danej klasy, które mogą uczestniczyć w asocjacji. Jest ona zapisywana przy końcach linii asocjacji na diagramie klas.

**Przykłady oznaczeń krotności:**

* **1** – dokładnie jeden obiekt (np. jeden użytkownik ma dokładnie jedno konto).
* **0..1** – zero lub jeden (np. pracownik może mieć jedno biurko lub nie mieć go wcale).
* **0..\* lub \*** – zero lub więcej (np. użytkownik może mieć dowolną liczbę zamówień).
* **1..\*** – co najmniej jeden (np. każdy zamówiony produkt musi być przypisany do co najmniej jednej kategorii).
* **2..5** – od dwóch do pięciu (np. drużyna może mieć od 2 do 5 członków).

Przykład:

1..\*

Klient ------------------------- Zamówienie

Jeden klient może mieć wiele zamówień, ale każde zamówienie musi należeć do jednego klienta.

W programie Visual Paradigm krotność pokazujemy poprzez podwójne kliknięcie na linię asocjacji, aby otworzyć okno edycji. Następnie w zakładce „Associaton Ends” znajdujemy pole „Multiplicity”, w którym możemy wprowadzić wartość krotności.

**Liczność** jest często używana zamiennie z krotnością, ale w ścisłym znaczeniu odnosi się bardziej do rzeczywistej liczby instancji obiektów w danym momencie działania systemu. Jest to pojęcie bardziej związane z implementacją i analizą konkretnej sytuacji w systemie.

Przykład:

Jeśli w bazie danych mamy 500 użytkowników i 2000 zamówień, to liczność relacji "użytkownik – zamówienie" w danym momencie to (500, 2000), ale jej krotność w modelu UML mogłaby być np. 0..\*.

Liczność możemy pokazać jako notatkę lub atrybut.

1. Asocjacja

**Asocjacja** to fundamentalna koncepcja, która opisuje związek między dwoma lub więcej klasami w systemie. Związki te wskazują, jak obiekty jednej klasy mogą współdziałać z obiektami innej klasy w danym systemie. Asocjacje pomagają w modelowaniu relacji między obiektami. Asocjacja reprezentuje logiczny związek między klasami, który może mieć różne właściwości. Każda asocjacja jest reprezentowana przez linię łączącą dwie klasy na diagramie klas UML. Asocjacja może mieć krotność, nazwę, nawiasy (np. 1..\*), role i inne cechy.

**Typy asocjacji**

**1. Asocjacja binarna (najczęstszy typ)**

* **Asocjacja binarna** łączy dokładnie dwie klasy.
* Reprezentuje relację między obiektami dwóch klas.
* Przykład: *Klient* składa *Zamówienie*.

**2. Asocjacja n-arna**

* **Asocjacja n-arna** obejmuje więcej niż dwie klasy.
* Nazywa się ją również **relacją wielu klas**.
* Przykład: *Zamówienie* jest związane z *Produktem* i *Specjalistą* w procesie realizacji.

Obraz zawierający tekst, diagram, zrzut ekranu, linia

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

**3. Asocjacja rekurencyjna**

* **Asocjacja rekurencyjna** występuje, gdy klasa jest powiązana ze sobą.
* Przykład: *Pracownik* może być **szefem** innego *Pracownika*.

Obraz zawierający tekst, Czcionka, zrzut ekranu, numer

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

1. Właściwość identyfikująca obiekty klasy (ograniczenie ID)

Każda klasa może mieć **właściwość identyfikującą obiekty** (ID), która jednoznacznie identyfikuje każdy obiekt danej klasy. Tego typu właściwość jest niezbędna do zapewnienia unikalności obiektów w systemie i pozwala na ich odróżnienie w ramach tej samej klasy. Jest to szczególnie ważne w kontekście systemów bazodanowych, systemów zarządzania danymi oraz wszędzie tam, gdzie wymagana jest identyfikacja obiektów w czasie działania systemu.

Właściwość identyfikująca może być wykorzystywana do tworzenia związków (asocjacji) między różnymi klasami. Dzięki unikalnemu ID możliwe jest łatwe identyfikowanie obiektów, które są powiązane w systemie.

Aby w Visual Paradigm oznaczyć atrybut jako unikalny wystarczy wejść w jego właściwości i zaznaczyć opcję ID.

1. Właściwość statyczna

W UML **właściwość statyczna** to atrybut lub metoda klasy, który:

* Należy do klasy, a nie do konkretnego obiektu.
* Jest współdzielony przez wszystkie instancje danej klasy.
* Może być wywoływany bez tworzenia obiektu klasy.

Przykład:  
maksymalnyCzasWyceny dla całej klasy taki sam

Aby w Visual Paradigm ustawić atrybut jako static należy w właściwościach atrybutu zmienić Scope z Instance na Classifier.