1. **W jakim celu buduje się diagramy klas?**
2. Wysokopoziomowa wizualizacja struktury systemu – daje wgląd w najważniejsze obiekty składające się na system, a także w ich cechy i zależności pomiędzy nimi.
3. Niskopoziomową (szczegółową) wizualizację struktury systemu – dokładnie opisuje strukturę systemu z uwzględnieniem szczegółów klas technicznych, obiektów ekranowych itp.
4. Wizualizację struktury bazy danych – przedstawia strukturę bazy danych.
5. Inżynieria wsteczna – wizualizacja struktury klas na podstawie kodu systemu napisanego w obiektowym języku programowania.
6. Definiowanie nowych języków – zawiera komplet pojęć wystarczający do definiowania gramatyki innych języków, zarówno tekstowych, jak i graficznych.
7. **Jaka jest procedura budowy diagramu klas?**
8. Identyfikacja obiektów ważnych dla projektowanego systemu.
9. Umieszczenie na diagramie najważniejszych (kluczowych) klas oraz powiązań pomiędzy nimi.
10. Umieszczenie na diagramie klas wspomagających klasy kluczowe.
11. Uzupełnienie diagramu o szczegóły.
12. **Z jakich elementów składa się klasa na diagramie klas?**
13. Symbole klas
14. Symbole atrybutów klas
    * Proste atrybuty
    * Złożone atrybuty
    * Wyliczenia
    * Kolekcje
15. Symbole operacji (metod) klas
16. Symbole interfejsów
17. Symbole oznaczające dziedziczenie
18. Oznaczenie typu klasy (np. abstrakcyjna)
19. Oznaczenie widoczności składowej diagramu (np. prywatna, publiczna)
20. Symbole związków pomiędzy klasami
21. Oznaczenia liczności związków między klasami
22. Oznaczenia ról związków między klasami
23. Zestawy ograniczeń / reguł poprawności dla składowych diagramu
24. **Jakie znasz rodzaje związków pomiędzy klasami?**
25. Dziedziczenie
26. Asocjacja
27. Agregacja
28. Kompozycja
29. Realizacja (implementacja)
30. Zależności
31. Znajomość
32. **Na czym polega agregacja?**

Agregacja odwzorowuje sytuację, w której agregaty składają się z obiektów innych klas tj. zależność część – całość, w której np. silnik jest składową samochodu.

1. **Co to jest agregat?**

Agregat – obiekt złożony z innych obiektów, gdzie obiekty składowe mogą istnieć niezależnie od agregatu.

np. Samochód jest agregatem, składającym się z części takich jak silnik, koła, drzwi itp. Te części mogą istnieć niezależnie od samochodu.

1. **Na czym polega kompozycja?**

Kompozycja jest odmianą agregacji, gdzie składniki agregatu (całości, kompozytu) nie mogą istnieć samoistnie i giną razem z agregatem np. człowiek ma serce.

1. **Co to jest kompozyt?**

Kompozyt to obiekt złożony z innych obiektów, gdzie części składowe nie mogą istnieć niezależnie od całości.

np. Dokument tekstowy – jest kompozytem składającym się z akapitów, tabel, rysunków.

1. **Jakie znasz elementy obiektowych języków programowania, które mają swoje odpowiedniki na diagramie klas?**
2. Klasy
   1. Symbole klas na diagramie
   2. Atrybuty klas
   3. Operacje (metody) klas
3. Dziedziczenie
   1. Relacje generalizacji/specjalizacji na diagramie
4. Interfejsy
   1. Symbole interfejsów na diagramie
5. Widoczność/dostępność składowych
   1. Oznaczenia widoczności np. publiczna, prywatna, chroniona
6. Asocjacje
   1. Relacje pomiędzy klasami na diagramie
   2. Liczności asocjacji
   3. Role asocjacji
7. Agregacja
   1. Relacja „część-całość” oznaczana pustym rombem
8. Kompozycja
   1. Specjalna agregacja, oznaczana wypełnionym rombem
9. Realizacja (implementacja)
   1. Relacja między klasą a interfejsem, oznaczana linią z pustym trójkątem
10. **Jakie typy danych są dostępne na diagramie klas?**
11. Proste typy danych:
    1. Liczby (int, float, double)
    2. Tekst (string)
    3. Logiczny (boolean)
    4. Data i czas
12. Złożone typy danych:
    1. Klasy (własne typy zdefiniowane w diagramie)
    2. Wyliczenia (enumeration)
    3. Struktury (structs)
13. Kolekcje:
    1. Listy
    2. Tablice
    3. Słowniki (mapy)
    4. Zbiory
14. **Jaka jest procedura tworzenia metody klasy?**

Zakładka „Operations” służy dodawaniu nowych metod klasy. Można w niej dokonać m.in. następujących ustawień:

1. Name – nazwa metody
2. Classifier – nazwa klasy
3. Return type – zwracana wartość
4. Visibility – widoczność metody
5. Body condition – miejsce na budowę ciała metody
6. Parameters – zakładka służąca dodawaniu parametrów metody
7. **Jakiego typu (rodzaju) mogą być parametry metod?**
   * 1. Proste typy danych:
        1. Liczby całkowite (int)
        2. Liczby zmiennoprzecinkowe (float, double)
        3. Znaki (char)
        4. Łańcuchy tekstowe (string)
        5. Wartości logiczne (boolean)
        6. Daty i godziny (Date, DateTime)
     2. Typy złożone:
        1. Klasy (własne typy zdefiniowane na diagramie)
        2. Wyliczenia (enumeration)
        3. Struktury (structs)
     3. Kolekcje:
        1. Listy (List)
        2. Tablice (Array)
        3. Słowniki (mapy)
        4. Zbiory (Set)
8. **Jak się określa liczność asocjacji pomiędzy klasami?**

Liczność asocjacji pomiędzy klasami na diagramie klas określa, ile obiektów jednej klasy może być powiązanych z obiektami innej klasy.

1. **Co można wygenerować na podstawie diagramu klas?**
   * 1. Definicje klas:
        1. Nazwy klas
        2. Atrybuty (pola) klas
        3. Metody (operacje) klas
     2. Hierarchie dziedziczenia:
        1. Relacje dziedziczenia między klasami
        2. Tworzenie klas pochodnych na podstawie klas bazowych
     3. Definicje interfejsów:
        1. Nazwy interfejsów
        2. Metody zdefiniowane w interfejsach
     4. Relacje między klasami:
        1. Asocjacje między klasami
        2. Agregacje i kompozycje
        3. Role i liczności asocjacji
     5. Konstruktory i inicjalizatory:
        1. Automatyczne generowanie konstruktorów na podstawie atrybutów klas
        2. Inicjalizowanie obiektów zgodnie z relacjami na diagramie
     6. Dostępność składowych:
        1. Modyfikatory widoczności (publiczne, prywatne, chronione)
        2. Generowanie odpowiednich modyfikatorów dostępu w kodzie
     7. Testowanie i dokumentacja:
        1. Tworzenie scenariuszy testowych na podstawie diagramu
        2. Generowanie dokumentacji projektowej i użytkownika
2. Co podlega dziedziczeniu pomiędzy klasami połączonymi generalizacją?
   * 1. Atrybuty (pola) klasy bazowej:
        1. Wszystkie publiczne i chronione atrybuty klasy bazowej są automatycznie dziedziczone przez klasy pochodne
        2. Prywatne atrybuty klasy bazowej nie są dziedziczone, ale mogą być dostępne przez metody publiczne/chronione
     2. Operacje (metody) klasy bazowej:
        1. Wszystkie publiczne i chronione metody klasy bazowej są automatycznie dziedziczone przez klasy pochodne
        2. Prywatne metody klasy bazowej nie są dziedziczone, ale mogą być wywołane przez metody publiczne/chronione
     3. Relacje z innymi klasami:
        1. Jeśli klasa bazowa ma relacje z innymi klasami, to te relacje są również dziedziczone przez klasy pochodne
     4. Ograniczenia i reguły:
        1. Wszelkie ograniczenia i reguły zdefiniowane na klasie bazowej są dziedziczone przez klasy pochodne