Inżynieria oprogramowania

Laboratorium nr 1. Podstawy UML, diagram klas

Zakres tematyki (3 x 45min):

- Wprowadzenie do **języka UML**
- Wstępne zapoznanie się z pakietem **StarUML** (charakterystyka, zakres funkcjonalności, najważniejsze opcje).
- Stworzenie **diagramów klas** w UML,
- generowanie kodu (tzw. stub code) na podstawie diagramu klas
- *Opcjonalnie: alternatywne metody modelowania klas w oparciu o schemat Code'a – Yourdone'a*

Język UML jest zunifikowanym językiem formalnym służącym opisu świata obiektów w szczególności do modelowania systemów informatycznych. UML jest zbudowany z elementów i przypisanych im symboli, które poprzez graficzną reprezentację, są wiązane ze sobą na diagramach. Choć UML był zaprojektowany, by definiować, wizualizować, konstruować i dokumentować systemy kładące nacisk na software, nie jest on ograniczony do modelowania oprogramowania. UML jest używany także do modelowania procesów biznesowych, inżynierii systemów i reprezentowania struktur organizacyjnych.

Literatura:
D.Pilone, N.Pitman *UML 2.0 Almanach* Wydawnictwo Helion
Booch i inni *Język UML* WNT

StarUML™ jest programistyczną platformą modelowania opartą na języku UML (Unified Modeling Language). Aktualna wersja (w listopadzie 2007) to 5.0.2.1270 bazuje na UML-u w wersji 1.4 wraz z rozszerzeniami zawartymi w wersji UML 2.0 (11 dostępnych typów diagramów). Platforma umożliwia wsparcie MDA (Model Driven Architecture).

Status programu StarUML™: freeware

Stąd można pobrać program StarUML™ (ver 5.0, 21.67Mb):
http://staruml.sourceforge.net/en/download.php

Dokumentacja developerska do programu StarUML™ (tylko w języku angielskim, .pdf 123 str): http://staruml.sourceforge.net/en/documentations.php

Budowanie modeli w StarUML™` Jak się nauczyć modelowania i zapisu w StarUML?
Help programu StarUML (bardzo dobry! – języku angielskim) - > spis treści

I. Budowanie modeli w StarUML podstawy:

I.1. Okno przy otwarciu projektu: New Project by Approach:

- Approach: wybór nowego projektu
  - **4+1 View model** - Model w postaci 4 widoków szczegółowych oraz 1-nego widoku ogólnego widoku przypadków użycia:
    - ***widok projektu*** (klasy, interfejsy, wzorce) Użycie diagramów klas, obiektów, aktywności, struktur złożonych i sekwencji
    - ***widok wdrożenia*** (konfiguracja, instalacja i wykonywanie) Komunikacja fizycznego układu sprzętu z systemem. Użycie diagramów komponentów, wdrożenia i interakcji.
    - ***widok implementacji*** (zarządzanie konfiguracją systemu, komponenty, wdrożenia, interakcje), diagramy komponentów oraz czasem interakcji, stanu i struktur złożonych
    - ***widok procesów*** ilustracja informacji nt. współbieżności, wydajności i skalowalności. Użycie diagramów interakcji i aktywności, które pokazują jak system zachowuje się podczas pracy
    - ***widok przypadków użycia*** – funkcjonalność wymagana systemu. Diagramy przypadków użycia + kilka diagramów interakcji pokazujących szczegóły tych pierwszych.
  - **Default approach, Rational Approach, UML Components Approach, Empty Project**
- Open Files: otwarcie pliku z wybranej lokalizacji
- Recent Files: zestawienie ostatnio otwartych projektów

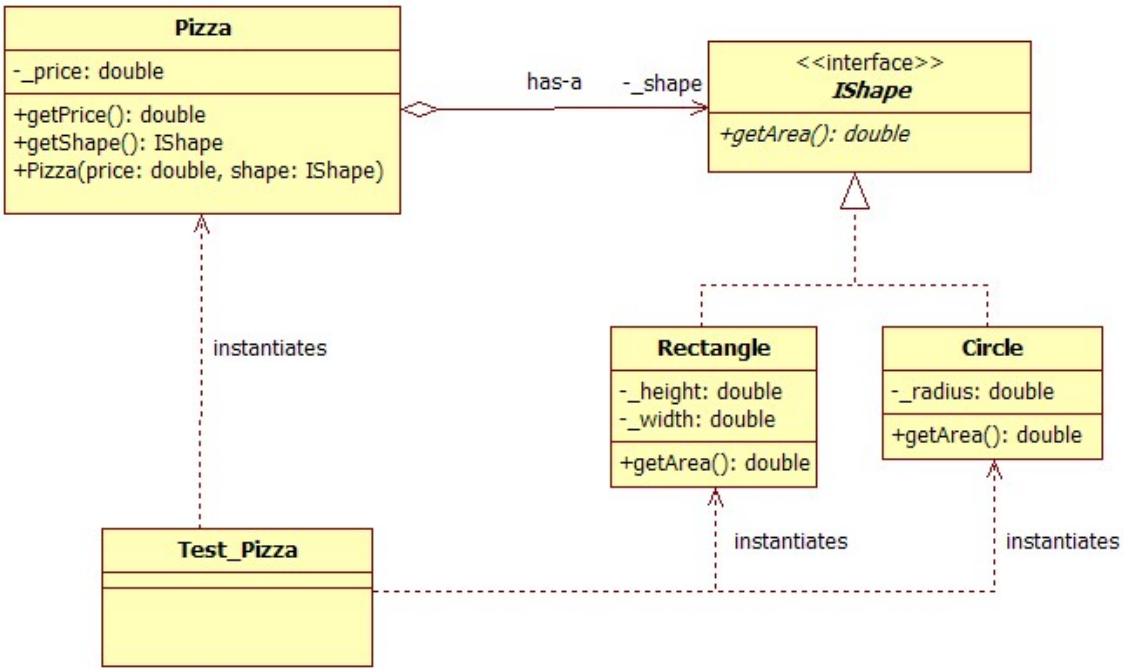
I.2. Tworzenie modelu na przykładzie diagramu klas:

- Wybierz menu **Model explorer/Diagram Explorer** lub **Diagram area** opcję utworzenia nowego diagramu danego typu
- Kliknij prawym przyciskiem myszy i wybierz opcję **Add diagram** z menu podręcznego. Wybierz **Class diagram.**
- Z menu **Toolbox** wybierz z diagramu (w tym przypadku diagramu klas) element który chcesz dodać (np. Class, Interface, Aggregation, Assotiation, itd.) Przeciągnij go do okna main
- Zdefiniuj parametry danego elementu (nazwa, widzialność itp) Okno **Properties** po prawej stronie u dołu. W przypadku klasy po dodaniu atrybutu zwróć szczególną uwagę na definicję typu – **Type.** Dodaj stosowane powiązania pomiędzy elementami.
- W przypadku usuwania danego elementu (np. klasy) lub powiązania wciśnij prawy przycisk na danym elemencie –uzyskasz menu podręczne następnie wybierz opcje **Edit** a dalej **Delete from Model (Ctrl+Del).** Niestety dany element z powiązaniami może figurować wciąż w tabeli powiązań: prawy przycisk myszy na obiekcie/**Collection Editor**/**Relations** zbędne powiązania trzeba tam usunąć ręcznie!

Uwaga: Jeżeli chcesz zmodyfikować pole atrybutu klasy np. typ. Kliknij na klasę, następnie wybierz w oknie **Properties** (po prawej stronie na dole) opcję **Atributes** dalej kliknij w opcję **…** i wybierz dany atrybut do modyfikacji.

II. Stwórz poniższy diagram klas UML

Celem tego ćwiczenia jest stworzenie za pomocą środowiska StarUML do stworzenia programu o nazwie Pizza. Wykonanie kolejnych kroków pozwoli na stworzenie diagramu UML pokazanego na rysunku poniżej. Przy pomocy StarUML-a dodatkowo wygenerujemy kod, który odzwierciedli strukturę klas, ale nie będzie specyfikował poszczególnych metod czy obiektów.*

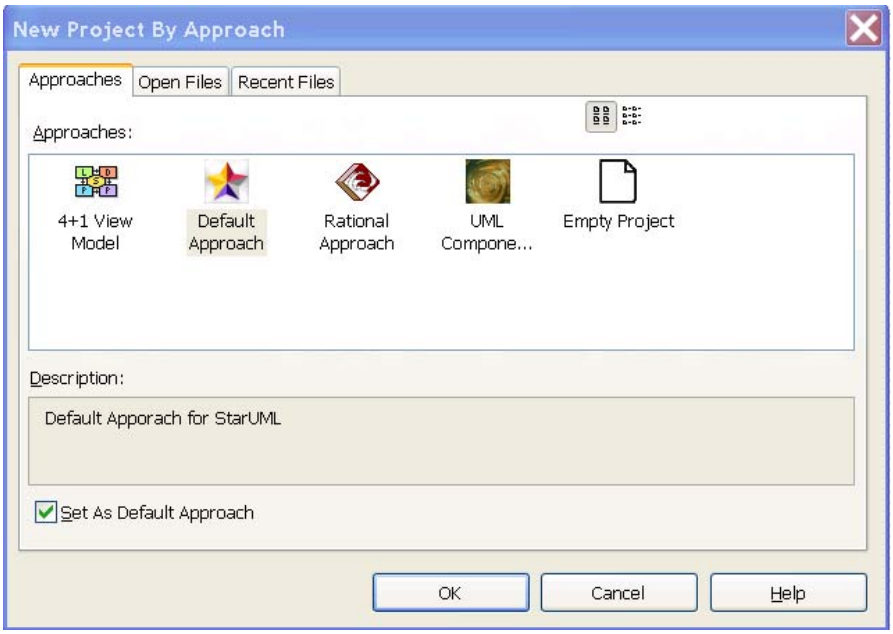


1. Instalacja: aby rozpocząć, musimy najpierw zainstalować program StarUML. Środowisko StarUML jest dostępne jako oprogramowanie open source zgodnie z zasadami GPL (GNU Public License), wpisując w dowolnej wyszukiwarce hasło „staruml dowload” uzyskamy linki do pobrania wersji instalacyjnej.

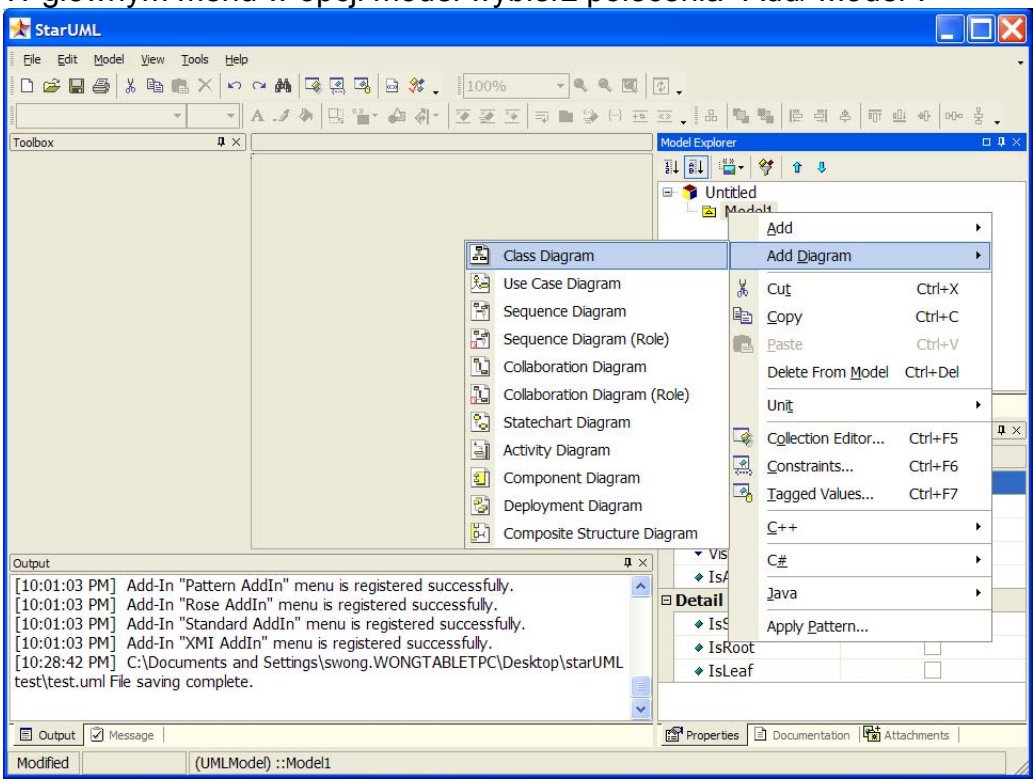
2. Gdy StarUML ("SU")jest zainstalowany proszę uruchomić program.

3. Po uruchomieniu SU, może wystapić formatka "New Project by Approach" jeżeli tak to wybierz "Empty Project" i potwierdź "Ok". Sugerowne jest odznaczenie opcji "Set As Default Approach".

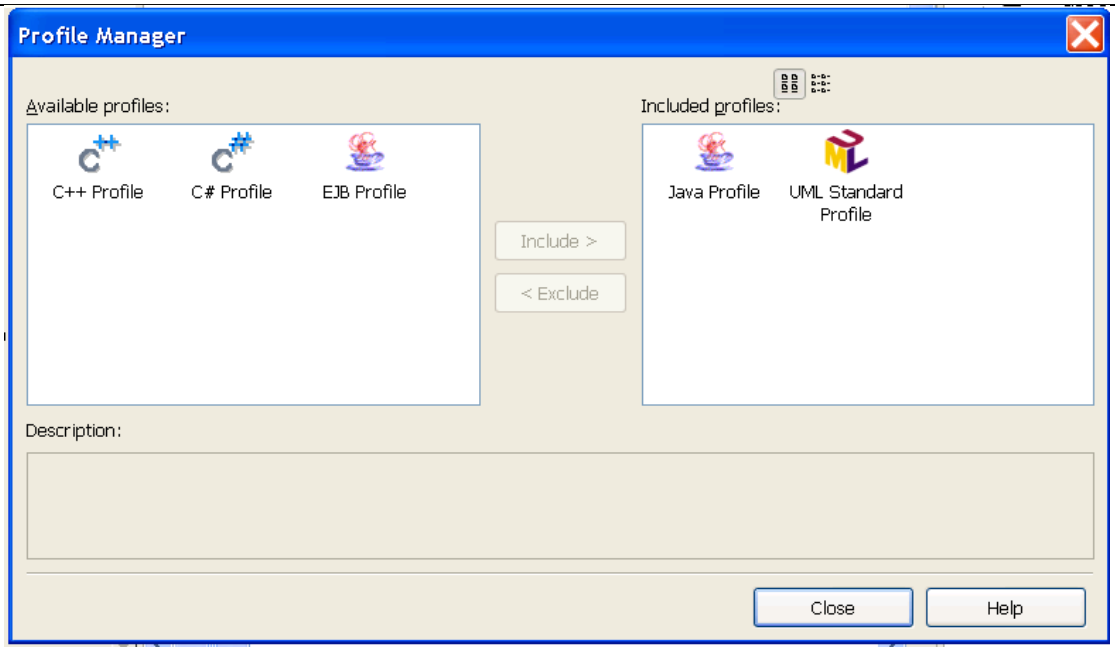
* - Aby utworzyć w pełni funkcjonalny program w obiektowym języku programowania należy na bazie uzyskanego szkieletowego kodu źródłowego tzw. „stub code” dokończyć tworzenie programu precyzując dokładnie, co każda z metod ma wykonywać.



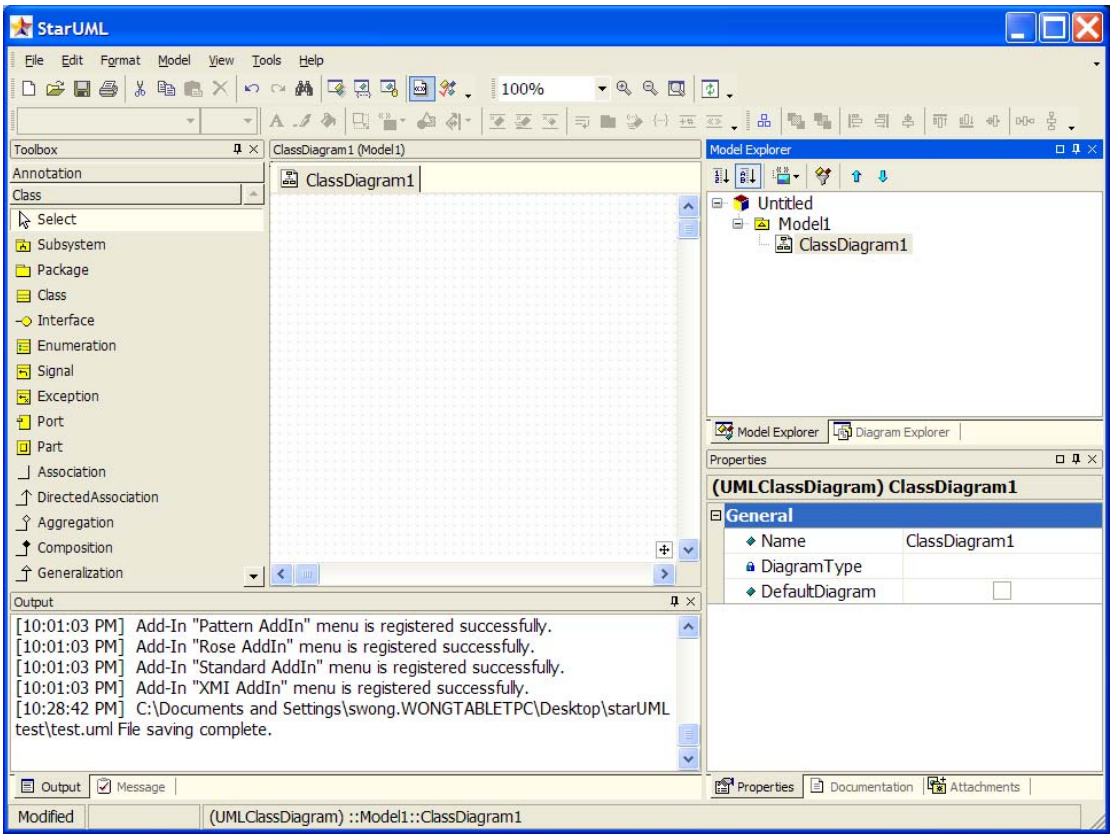
4. W panelu "Model Explorer" umiejscowionym po prawej stronie w górze, powinien być wybrany "Untitled" model .
5. W głównym menu w opcji model wybierz polecenia "Add/ Model".



6. Przejdź do "Model/Profile..." aby ustawić profile używane w projekcie, które determinują użyte symbole i konwencje. Użyj profil **"C++"** w projekcie.



7. Zapisz teraz projekt, aby w razie problemów nie stracić danych. Z menu "File" wybierz opcję "Save", następnie wybierz lokalizację aby zapisać projekt. Twój projekt powinien teraz wyglądać następująco:

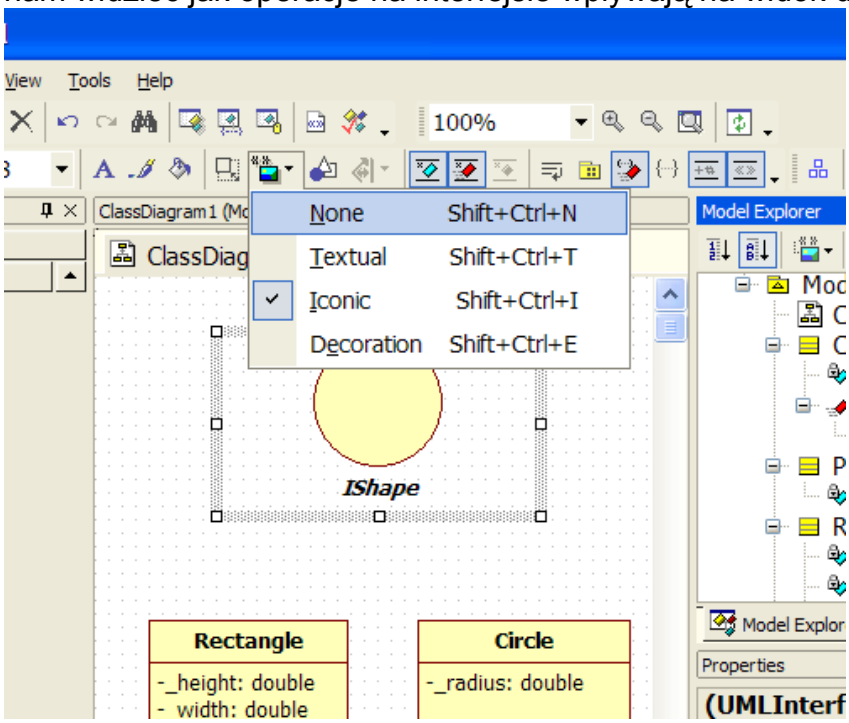


8. Teraz, aby rozpocząć tworzenie diagramów z menu "Toolbox", które jest otwarte domyślnie, wybierz ikonę "Class" i kliknij lewym przyciskiem myszy w dowolnym miejscu okna „ClassDiagram”. Tym sposobem tworzona jest klasa z pewną domyślną nazwą. Zmień nazwę klasy na **`Circle`** poprzez podwójne kliknięcie na domyślną nazwę.

9. Dodaj atrybut "Attribute" do klasy **`Circle`** poprzez kliknięcie prawym przyciskiem myszy obiektu na diagramie, poprzez rozwinięcie menu "Add" i wciśnięcie zielonego przycisku "Attribute". Wpisz nazwę pola "`_radius`".
   - Określ typ danych w panelu właściwości “Properties panel” (dolny, rawy fragment okna) poprzez wybranie typu **`double`** w polu "Type". ***Uwaga!: pole „Type” wyświetla się automatycznie tylko przy utworzeniu atrybutu (przy pierwszym uruchomieniu). Aby zmodyfikować to pole później, będąc w danej klasie (menu properties) należy kliknąć w pole „Attributes” ->”Collections”, i wybrać kliknięciem dany atrybut, aby zdefiniować jego zwracany typ.***
   - Wewnętrzne dane klasy pola i atrubuty (field/attributes) są zawsze definiowane jako prywatne, ponieważ przeznaczone są wyłącznie do użycia przez klasę. Dlatego też w panelu właściwości dla pola “**`_radius`**” wybierz zakres widzialności jako PRIVATE!

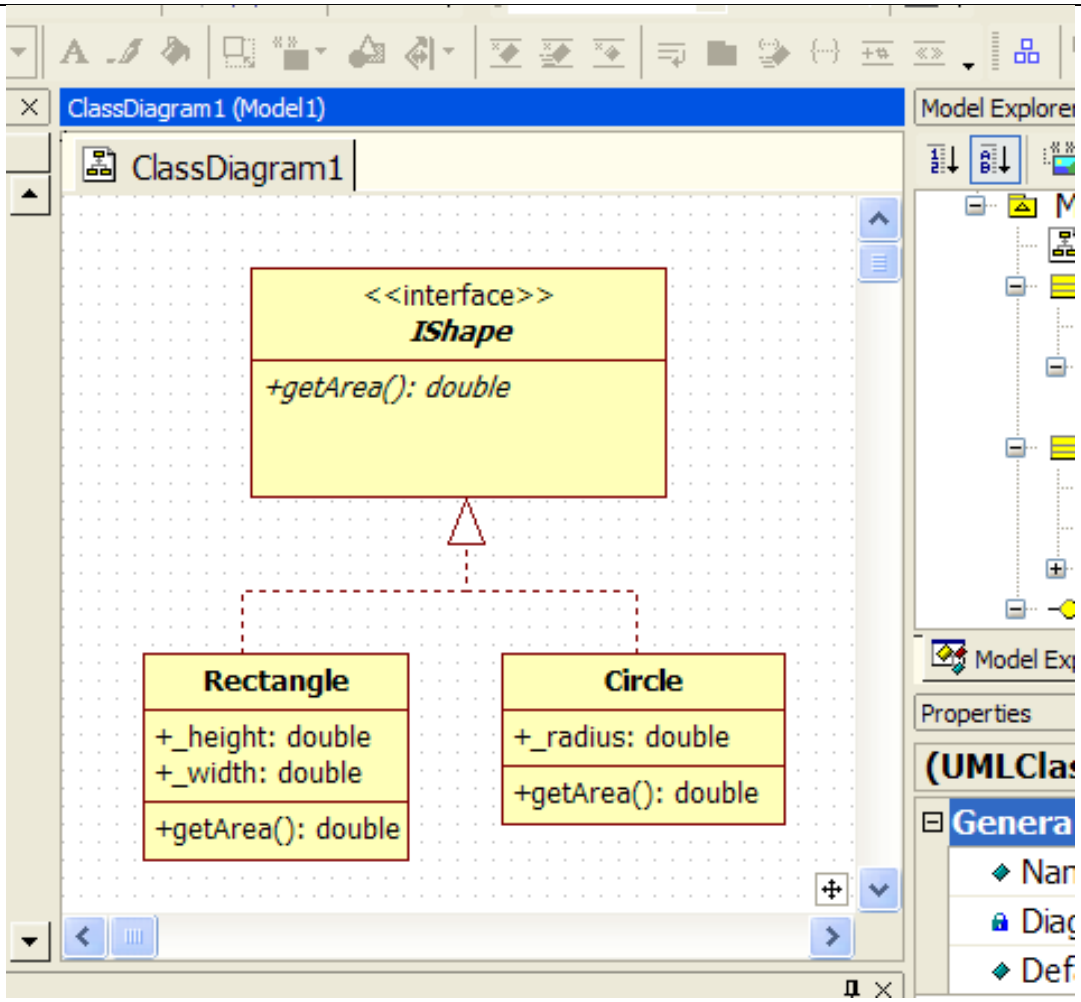
10. Powtórz proces tworzenia klasy nazwanej *Rectangle* z prywatnymi składowymi: **`_width`** and **`_height`** typu **`double`**.

11. Utwórz interfejs o nazwie **`IShape`**
    - Z menu Toolbox wybierz "Interface" i kliknij w dowolnym miejscu palety. zmień domyślną nazwę na **`IShape`**. Wybierz interfejs przez kliknięcie lewym przyciskiem na utworzony element.
    - W głównym menu wybierz opcję "Stereotype Display" i zmień jej wartość na "None". To spowoduje zmianę okrągłego kształtu na prostokątny.
    - w tym samym menu, odznacz opcję "Suppress Operations". To pozwoli nam widzieć jak operacje na interfejsie wpływają na widok diagramu.



- Dodaj metodę **`getArea`** typu **`double`** do interfejsu **`IShape`**.
  - Należy to wykonać poprzez kliknięcie interfejsu prawym przyciskiem myszy rozwinięciem menu “add” i wciśnięciem czerwonej ikonki "Operation" . Wprowadź nazwę operacji: **`getArea`**.
  - Aby ustawić zwracany typ rozwiń element **`IShape`** w oknie **"Model Explorer"** (po prawej stronie ekranu), klikając prawym przyciskiem na metodę **`getArea`**, którą właśnie stworzyłeś i wybierając opcję "Add Parameter". W formatce "Properties" , zmień nazwę parametru na pustą, "", zmień opcję "DirectionKind" na "RETURN", a następnie zmień "Type" na **`double`**.
- Na interfejsie **`IShape`** oraz metodzie **`getArea`** zaznacz opcję “IsAbstract” w panelu właściwości. Wówczas zgodnie z notacją UML ich nazwy będą wyświetlane kursywą jako czysto abstrakcyjne (ang. purely abstract).

12. Zaimplementuj klasy **`Circle`** and **`Rectangle`** za pomocą **`IShape`** poprzez wybranie strzałki "Realization" z menu “toolbox”, następnie kliknij na **`Circle`** i przeciągnij linię do **`IShape`**. Powtórz ten sam proces dla klasy **`Rectangle`**. Dodanie takiej relacji spowoduje, że klasy **`Circle`** and **`Rectangle`** będą implementowały interfejs **`IShape`**.
    - Aby stworzyć ładnie sformatowaną linię łączącą kliknij prawym przyciskiem myszy na linię i wybierz "Format/Line Style/Rectilinear". Możesz poprawić przejrzystość swojego diagramu poprzez połączenie grotów strzałek na diagramie, tak aby gdzie to możliwe był widoczny jeden grot strzałki zamiast kilku.
13. Od kiedy klasy **`Circle`** oraz **`Rectangle`** implementują interfejs **`IShape`**, muszą mieć zdefiniowane pewne metody jako **`IShape`**.
    - W panelu “Model Explorer” skopiuj metodę **`getArea`** (Ctrl-C lub prawy przycisk i opcja “Copy”) z klasy **`IShape`** do klas **`Circle`** oraz **`Rectangle`**.
    - Zaimplementowane metody w klasach **`Circle`** oraz **`Rectangle`** nie są abstrakcyjne, ale konkretne ponieważ aktualnie wykonują pewne działanie (przykładowo liczą pole koła lub prostokąta). Tak więc pole “IsAbstract” musi być *odznaczone* dla tych metod.
14. Twój diagram powinien wyglądać mniej więcej następująco:

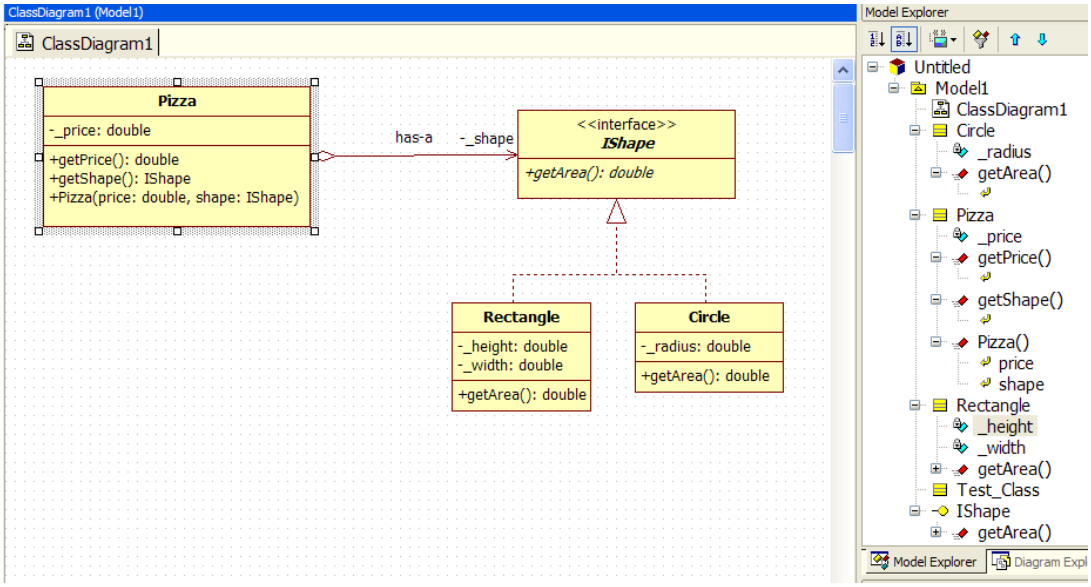


15. Dodaj do diagramu klasę o nazwie **`Pizza`**.
    - Dodaj prywatną składową typu **`double`** o nazwie **`_price`**.
    - Dodaj publiczną metodę **`getPrice`**, która zwróci typ **`double`**.

16. Aby utworzyć odwołanie z klasy **`Pizza`** do klasy **`IShape`**, wybierz klasę **Pizza**:
    - Wybierz strzałkę "DirectedAssociation" w menu toolbox, kliknij na klasę **Pizza** i przeciągnij do **`IShape`**.
        - Teraz wybierz strzałkę i w menu "Properties" po prawej, zmień nazwę na "has-a", zmień również "End1.Aggregation" na "AGGREGATE" (to formalna gramatyczna deklaracja, że pizza jest „przygotowana” tzn. "zagregowana", z innym obiektem w tym przypadku obiektem kształtu (shape object)).
        - Zmień "End2.Name" na _shape. To spowoduje automatyczne dodanie składowej prywatnej o nazwie **`_shape`** typu **IShape** do klasy **Pizza**.
        - Zmień widzialność End2. na PRIVATE.
    - Utwórz metodę (operację) obsługującą **`_shape`** o nazwie **`getShape`**, która zwraca **`IShape`** (**zwracany typ zmiennej IShape wpisz ręcznie w nawiasie**).

17. Konstruktory są specjalnymi elementami kodu, używanymi do inicjalizacji instancji (inaczej wystąpienia obiektu) klasy. Możesz opcjonalnie stworzyć konstruktory w klasie **Pizza**.
    - Aby dodać konstruktor dla klasy **`Pizza`**, przyciśnij prawym przyciskiem myszy na klasie **`Pizza`**, rozwiń menu "Add" menu i wybierz "Operation". Teraz wprowadź nową metodę (operację) z wejściowymi parametrami parametrami double **`price`** oraz IShape **`shape`**.
        - Dodanie parametrów wejściowych jest analogiczne jak dodawanie parametrów wyjściowych

18. Twój diagram powinien wyglądać następująco:



19. Aby zilustrować kolejne cechy diagramu klas w UML dodaj kolejną klasę o nazwie **Test_Pizza** do twojego diagramu. To będzie klasa, która używa klas pochodnych **Pizza** oraz **IShape** do celów testowych.
    - Linie zależności (dependency lines) pomogą pokazać relacje pomiędzy klasami (które mogą wystąpić dynamicznie). Na przykład, jedna klasa może wpływać na tworzenie obiektów w innej klasie bez przechowywania stałego odwołania przez użycie pola. Lub metoda klasy może użyć innej klasy jako wejściowego parametru poprzez przechowywanie odwołania do niej tylko jako podczas trwania wykonywania tej metody.
    - Dodaj zależności pomiędzy kilkoma różnymi klasami poprzez wybranie strzałki "Dependency" z menu Toolbox, wybierając zależną klasę i przeciągając strzałkę do klasy ”nadrzędnej”. W naszym przypadku, **Test_Pizza** zależy od klas „nadrzędnych” **Pizza**, **Circle** oraz **Rectangle** ponieważ powoduje tworzenie konkretnych obiektów.

- Zdefiniuj etykietę (label) dla zależności zaznaczając własność "Name" w menu „Properties” lub poprzez podwójne kliknięcie w linię zależności (dependency line). Typowo kiedy jedna z klas powoduje wystąpienie obiektów innej nazywamy etykietę "instantiates".
- Możesz przesunąć etykietę linii zależności w inne miejsce przez zaznaczenie jej i przeciągnięcie w inne miejsce.
- Zależności (Dependences) nie mają wpływu na wygenerowany kod.

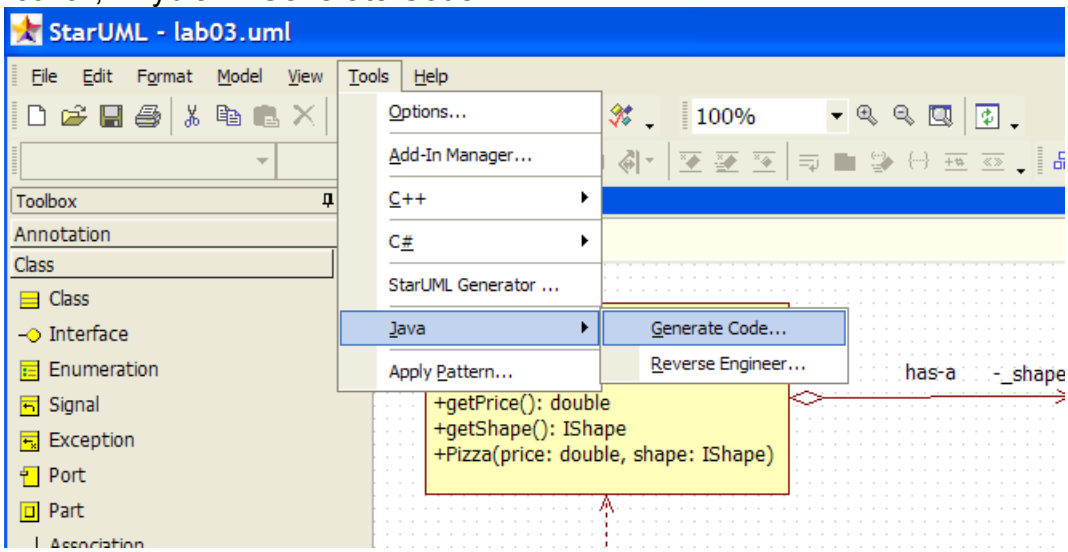
20. Twój diagram powinien wyglądać podobnie jak ten na pierwszym rysunku.

21. Możesz modyfikować dalej swój diagram (np. przeciągać diagramy klas w dowolne miejsce okna). Spróbuj poeksperymentować z formatowaniem diagramów i narzędziami.

22. W menu File, wybierz Save. SU używa pojedynczego pliku projektu dla zebrania wszystkich danych (plik ten ma rozszerzenie .uml).

23. Bardzo pożyteczną opcją jest eksport diagramów do innych formatów (jako obrazki). Możesz to zrobić poprzez wybór "Export Diagram" w menu File i wybraniu właściwego typu pliku docelowego.

24. Aby wygenerować szkielet kodu (ang. stub cod) np C++ lub Java:
    - Przejdź do menu "Tools"w głównym menu, rozwiń opcję “C++” lub "Java", i wybierz "Generate Code".



    - Z tego okna dialogowego wybierz Twój model prawdopodobnie nazwany "Model1" i wciśnij "Next"
    - Wybierz opcję "Select All" aby wygenerować “stub code” dla wszystkich klas Twojego modelu i wciśnij "Next".
    - Wybierz katalog do zapisu i wciśnij "Next"
    - W otwartym oknie "Options Setup",
        - w przypadku C++ nie zapomnij o zaznaczeniu opcji „Generate documentation to comment”
        - w przypadku Javy upewnij się czy wybrałeś "Generate the Documentation by JavaDoc" oraz "Generate empty

JavaDoc". pozostałe opcje powinny być niezaznaczone. Następnie wciśnij "Next".

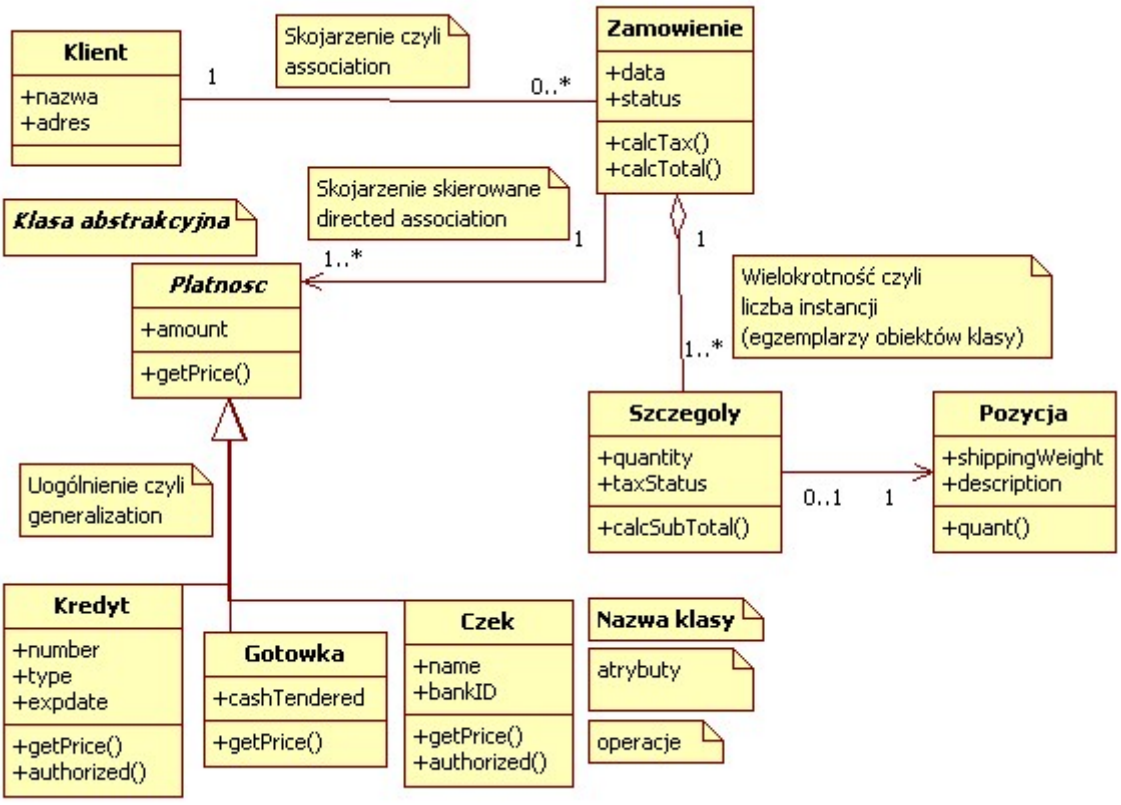
- StarUML wygeneruje szkielet kodu z Twoich diagramów. Kliknij opcję "Finish" aby opuścić okno dialogowe.
- teraz można edytować uzyskany kod i skończyć edytować aplikację

Ostrzeżenie o błędzie: w wersji 5.0.2.1570 programu StarUML, w sytuacji kiedy linia generalizacji “generalization line” zostaje usunięta z diagramu, może nie być równoczesnie usunięta z modelu. To może spowodować stworzenie niepoprawnego kodu. Aby wykryć czy są wciąż powiązania pomiędzy klasami wykonaj następujące czynności:

1. Kliknij prawym przyciskiem na klasę i wybierz "Collection Editor".
2. W menu “Collection Editor” wybierz tabelę relacji "Relations".
3. tabela relacji pokaże wszystkie relacje związane z daną klasą (zarówno “wychodzące” i “przychodzące”)
4. Jeżeli jest więcej relacji pokazanych w tabeli relacji niż na diagramie sprawdź która relacja jest błędna.
5. Aby usunąć relację nie widoczną w diagramie klas wciśnij prawy przycisk na tabeli relacji i wciśnij "Delete".

III. Stwórz w języku UML diagram klas jak na rysunku poniżej

Nie zapomnij o uwzględnieniu zawartych notek! Na poniższym rysunku prostokąty z komentarzem (wizualnie z zawiniętym prawym górnym rogiem)



Niniejszą instrukcję i zawarte w niej przykłady przygotowano w oparciu o następującą literaturę:

1. Pilone, Pitman *UML 2.0 Almanach* Wydawnictwo Helion 2007
2. Booch G. (i inni) *UML Przewodnik użytkownika inżynierii oprogramowania* Wydawnictwo WNT 2002.
3. Muller R. *Język UML w modelowaniu danych* Mikom 2000
4. Minkyu Lee i inni *StarUML User Guide* 2005
    a. http://staruml.sourceforge.net/docs/user-guide(en)/toc.html
5. StarUML Tutorial
    a. http://owlnet.rice.edu/~comp201/07-spring/info/staruml/
6. Miller R. Język UML w praktyce: wprowadzenie dla programistów
    a. http://www.borland.pl/tech/poradnik_uml.shtml
7. Erudis Przegląd diagramów języka UML
    a. http://www.erudis.pl