Ministerul Educaţiei al Republicii Modova

Universitatea de Stat din Moldova

Facultatea de Fizica si Inginerie

Catedra Fiziсă Aplicată şi Informatică

Lucrare de laborator Nr. 2

Tema: **“ *Diagrama Claselor*”**

Lucrarea a fost îndeplinită de studentul grupei 3.2:

Ciobanasu Ion

Lucrarea a fost controlată de

lector univ: Diaconu Elena

Chisinau 2016

**Note teoretice:**

Diagrama claselor este un tip de diagramă utilizată pentru descrierea structurii statice, adică a entităților sau claselor existente într- un sistem.

Acest tip de diagramă este utilizat cel mai des de către dezvoltatori pentru specificarea claselor, dar poate fi util și pentru specificarea structurii unei sisteme sau unei subsisteme dintr- un bussiness real. Diagramele claselor sunt structuri statice, deoarece nu descriu ceea ce se întâmplă în momentul în care clasele relaționate interacționează. Principalele proprietăți ale unei clase sunt: numele, stereotipul și vizibilitatea.

Pentru a crea o diagramă a claselor, trebuie să identificăm clasele și apoi să le descriem. Clasa în limbajul UML definește totalitatea de obiecte care au aceeași structură, comportament și relații cu obiectele din alte clase.

Identificarea claselor se realizează răspunzând la următoarele întrebări:

1. Există un tip de informație care trebuie de depozitat?
2. Există sisteme externe implicate în activitatea sistemului nostru?
3. Există dispozitive pe care sistemul nostru trebuie să le gestioneze?
4. Există biblioteci sau componente din alte proiecte, care vor fi folosite în sistem?
5. Actorii folosiți au un rol în sistem, adică vor trebui implementați ca fiind clase?

**Tipuri de clase**

**I.Clase active-**inițiază și controlează fluxul activității.

**II.Clase pasive-**depozitează informația și servesc celorlalte clase.

**III.Clase abstracte-**nu pot fi instanțiate obiecte de tipul acestor clase.

**IV.Clase concrete-**opusul celor abstracte.

O clasă este reprezentată printr-un dreptunghi cu 3 nivele. În cle de sus se indică numele clasei, în cel de mijloc se indică atributele clasei și în al treilea nivel- operațiile specifice clasei.

Semnul minus indică faptul că atributele sau funcțiile sunt de tip *private,* iar semnul plus – tipul de acces *Public* pentru atribute și funcții. După specificatorul de acces, urmează denumirea atributelor și funcțiilor separate prin simbolul ”:” de tipul de date al atributului sau de tipul de date returnat de funcție.

O clasă abstractă va avea numele scris cu caractere italice și va fi un substantiv ce se referă la domeniul problemei analizate. Atributul clasei se folosește pentru a păstra informații care descriu și identifică o caracteristică specifică a clasei. Un atribut are un tip definit: ( real, integer, string, byte, char) sau poate fi o instanță a unei clase construite.

Declarațiile de acces sunt:

*Public (* + *)*- poate fi accesat, văzut din altă clasă.

*Private (* - *)-* nu poate fi accesat sau văzut din alte clase.

*Protected (* # *)* – acceptă referirea numai la clasele ”fiu” moștenitoare.

Operațiile sunt folosite pentru a manipula atributele sau pentru a realiza alte acțiuni. Sunt, de obicei, numite funcții sau metode. Au un nume, un tip returnat și 0 sau mai mulți parametri. Operațiile într- o clasă descriu ce poate face o clasă, ce servicii oferă, prezentând fiecare exemplar al clasei după o anumită cerință.

Tipuri de relații între clase:

* **Relații de asociere-** se reprezintă printr- o linie cu simboluri speciale care caracterizează unele proprietăți ale asocierii concrete sau cu o săgeată orientată- în cazul în care asocierea e cunoscută doar de una dintre clase. Spre deosebire de diagrama USE- CASE, într- o diagramă de clasă mai apare așa- numita asociere recursivă, adică o clasă poate fi conectată cu ea însăși printr- o asociere. De exemplu:
* **Relații de moștenire-** relații care pot apărea între interfețe sau clase, însă nu și între interfețe și clase. Reprezentarea este printr- o linie cu un triunghi alb la capăt spre clasa de bază.
* **Relații de dependență-** este o relație semantică între 2 elemente, modele sau între 2 mulțimi ale acestor elemente, unul fiind dependent de celălalt, care e independent. O schimbare în elementul independent îl va afecta și pe elementul dependent.
* **Relații de agregare-** există între 2 clase în cazul când o clasă reprezintă o oarecare entitate, care include în sine, în calitate de părți componente alte entități. Se reprezintă printr- o linie cu un romb gol la capătul dintre superclasă.
* **Relația de compoziție-** este un caz particular al relației de agregare și se folosește pentru o formă specială de relații: component întreg *component- întreg* , în care componentele aparțin unui întreg. Interacțiunea între ele constă în aceea că părțile nu pot exista independent, adică cu distrugerea clasei principale se vor distruge toate părțile componente.
* **Relația de generalizare-** este o relație taxonometrică între 2 elemente de același tip, asemănătoare cu cea de moștenire.

*Interfețe*

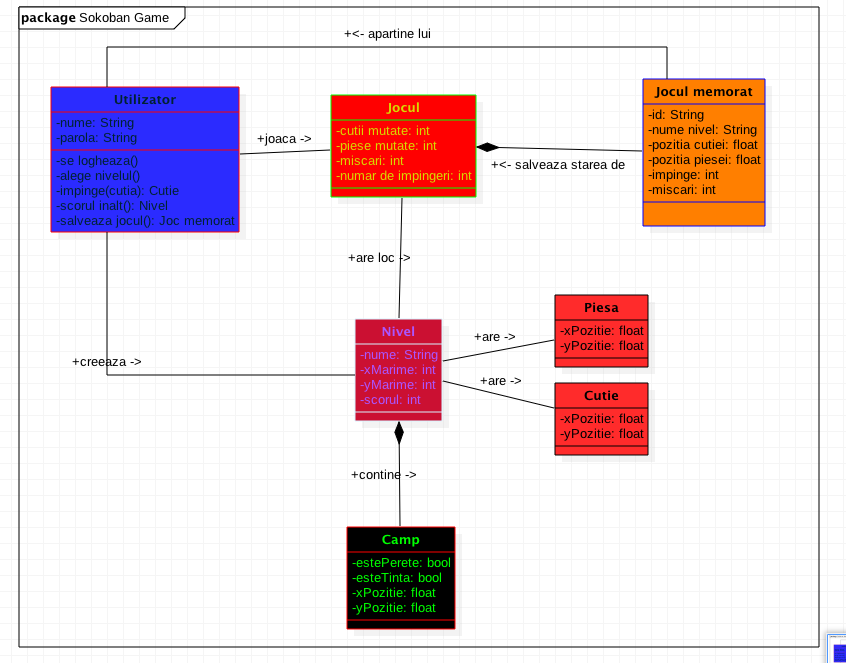
Interfețele sunt un fel de clase care conțin doar operații, nu și atribute. Grafic, o interfață arată ca un dreptunghi cu 2 compartimente: numele și operațiile, având stereotipul

<< interface >>. Operațiile sunt abstracte și nu au nici o implementare în cadrul interfețelor, însă clasele care au o legătură cu o interfață, vom spune că implementează acea interfață, aplică operațiile și metodele.

Obiectul este un exemplar special al clasei, care este creat în timpul executării programului. El are un nume propriu și valoare concretă a atributelor. Dacă numele clasei lipsește, se indică numele obiectului. Atributele obiectelor primesc valori concrete.

**PARTEA PRACTICĂ**

*De elaborat un sistem de clase pentru joaca logica Sokoban*

**

***Concluzie:*** Efectuînd lucrarea de laborator am cunoscut ce e diagrama claselor, importanța acestui tip de diagramă și cum să creează diagrama claselor. Definirea claselor este asemănătoare cu definirea claselor din limbajele de programare orientată pe obiecte. Astfel, avînd cunoștințe în POO , nu va fi foarte dificil de creat clase pentru diagramă.

Ca și clasele din programare, clasele definite în UML au atribute și metode de acces. Acestea pot fi publice, adică accesibile obiectelor din alte clase, private- accesibile doar cu ajutorul unor metode ( funcții ) definite de către programator, protected- accesibile doar obiectelor din clasele moștenitoare. Diagrama claselor ajută la reprezentarea entităților din lumea reală și a dependențelor dintre acestea, într- o formă schițată ce poate fi explicată uneori mai simplu, decît prin limbaj natural.