**Ministerul Educatiei Republicii Moldova**

**Universitatea Tehnica a Moldovei**

**Departamentul ISA**

**Raport**

La lucrarea de laborator № 1

Disciplina “Analiza, proiectarea și programarea obiect orientata”

**Tema: Implementarea principiilor de baza POO**

A efectuat: st. gr. TI-153 Suruceanu V.

A controlat: prof. univ. Pecari M.

Chisinau 2017

**Scopul Lucrarii:** Implementarea principiilor de baza POO

**Despre**

**Programarea orientată pe obiecte** (POO, uneori și **Programarea orientată obiect**, uneori denumită ca și în limba engleză, **Object Oriented Programming (OOP)**) este o paradigmă de programare, axată pe ideea încapsulării, adică grupării datelor și codului care operează asupra lor, într-o singură structură. Un alt concept important asociat programării orientate obiect este polimorfismul, care permite abstractizări ce permit o descriere conceptuală mai simplă a soluției.

* **Abstractizarea** – Este posibilitatea ca un program să ignore unele aspecte ale informației pe care o manipulează, adică posibilitatea de a se concentra asupra esențialului. Fiecare obiect în sistem are rolul unui “actor” abstract, care poate executa acțiuni, își poate modifica și comunica starea și poate comunica cu alte obiecte din sistem fără a dezvălui cum au fost implementate acele facilitați. Procesele, funcțiile sau metodele pot fi de asemenea abstracte, și în acest caz sunt necesare o varietate de tehnici pentru a extinde abstractizarea:
* **Încapsularea** – numită și *ascunderea de informații*: Asigură faptul că obiectele nu pot schimba starea internă a altor obiecte în mod direct (ci doar prin metode puse la dispoziție de obiectul respectiv); doar metodele proprii ale obiectului pot accesa starea acestuia. Fiecare tip de obiect expune o interfață pentru celelalte obiecte care specifică modul cum acele obiecte pot interacționa cu el.
* **Polimorfismul** – Este abilitatea de a procesa obiectele în mod diferit, în funcție de tipul sau de clasa lor. Mai exact, este abilitatea de a redefini metode pentru clasele derivate. De exemplu pentru o clasă Figura putem defini o metodă arie. Dacă Cerc, Dreptunghi, etc. ce vor extinde clasa Figura, acestea pot redefini metoda arie.
* **Moștenirea** – Organizează și facilitează polimorfismul și încapsularea, permițând definirea și crearea unor clase specializate plecând de la clase (generale) deja definite - acestea pot împărtăși (și extinde) comportamentul lor, fără a fi nevoie de a-l redefini. Aceasta se face de obicei prin gruparea obiectelor în clase și prin definirea de clase ca extinderi ale unor clase existente. Conceptul de moștenire permite construirea unor clase noi, care păstrează caracteristicile și comportarea, deci datele și funcțiile membru, de la una sau mai multe clase definite anterior, numite *clase de bază*, fiind posibilă redefinirea sau adăugarea unor date și funcții noi. Se utilizează ideea: ”Anumite obiecte sunt similare, dar în același timp diferite”. O clasă moștenitoare a uneia sau mai multor clase de bază se numește *clasă derivată*. Esența moștenirii constă în posibilitatea refolosirii lucrurilor care funcționează.

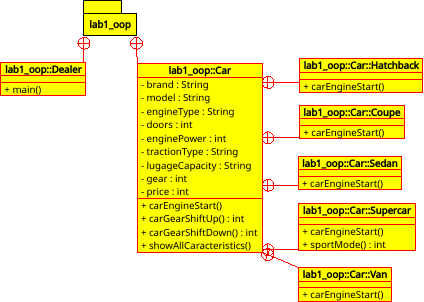


Figura 1 – Class diagram

**Concluzie**

In cadrul acestui lucrari de laborator au fost invatate principille de design a softurilor programarea orientată pe obiecte. Cunostintele obtinute au fost sistematiza in raportul dat, care in viitor va fi folosit in scopuri practice, ca referinta.