**07.06.2023**

1. Recap. Abstractizare?

2. Object Oriented Technology -**OOT** -(a inglobat multe aplicatii din lumea actuala)

- requirements -> Descriere App. (comportament, structura)

HTML – hypertext markup language

HTTP(s) – hypertext transmission protocol

IDE – Integrated development environment

URL – Unified Resource Locator

Functionalitate = actiune (verb)

**OOT** – is a software design model in which objects contain both data and the instructions that work on the data. It is increasingly deployed in distributed computing. Obiecte / o retea distribuita, cu informatii si functionalitati.

Aplicatie observatii:

* NU descrierea utilitatii aplicatiei, ci a functionalitatii ei
* Ai descris o dubla-functionalitate a aplicatiei
* Lipsa baza de date unde se salveaza user si parola
* Formularea este ambigua cand dai optiunea de a se loga ca si vizitator, sau ca user inregistrat. Pune accentual pe vizitator (nu e suficient de explicit aplicatia nu stie ce inseamna vizitator); ce face aplicatia in situatia asta? Trimite un cod de acces pe telefon (sau cum se face conexiunea?)
* Aplicatia va genera un desktop virtual si va afisa aplicatiile instalate pe telefon sub o alta forma decat cum sunt pe telefon, dar sub forma de iconite
* Restrange functionalitatile aplicatiei (gen ai acces doar la apeluri pierdute si SMS-uri)

**Descrie ce face aplicatia?**

**13.06.2023**

1. Recap. OOT – designul softwarelului, arhitectura orientate pe obiect, are legatura cu internetul care are resurse (inmagazineaza instructiuni care ruleaza aceste date);

- contine date, informatie in obiect si functionalitati care utilizeaza aceste date;

- Obiecte care vor comunica intre ele, date si instructiuni;

- Datele sunt inmagazinate, functionalitatile sunt instructiuni care folosesc aceste date.

OOT sta la baza a ceea ce fac designerii si a celor care fac OOP

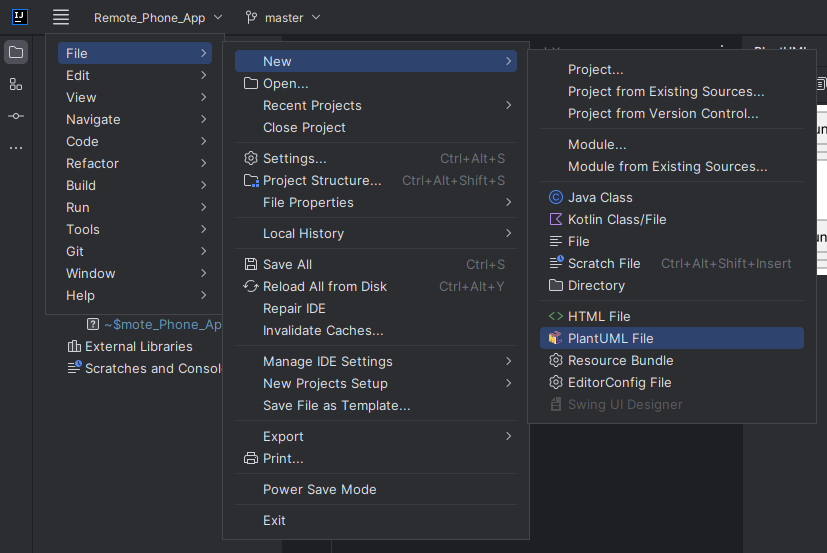
2. **UML** (requirements) - **Unified Modeling Language**

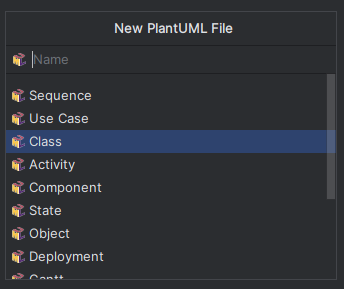
Este un limbaj standardizat pentru modelare. Ajuta developerii sa vizualizeze, sa construiasca si sa documenteze noi sisteme software si diagrame. Contine un set de reguli specifice pentru creearea diagramelor.

Posibilitatea de a creea diagrame (**class diagram** / **use case diagram**) diagramele cu care ne vom forma structura.

* Use Case diagram (diagrama de comportament) – defineste cateva elemente:
  + Behavior - Who and What, not ~~How~~ is happening
    - UC diagram (cazuri de utilizare a aplicatiei)
      * Utilizare = functionalitate (actiuni/verbe)

Actor (subiect) joaca un rol (ex. utilizator) <–> Use case (cauta ce actiuni se vor genera) – descriere a cazului de folosinta (ex. afisarea ferestrei de login)





**22.06.2023**

Use Case (UC) – Diagrama de comportament (functionalitati)

DIf intre clase si obiecte – proprietatile au valoare;

Prin clase vei stabili comportamentul use case-urilor.

Prin cele 3 tipuri context, compozitie si stare stabilim….

Aggregate (care depind de alte obiecte) and Composite (care nu depind de alte obiecte)

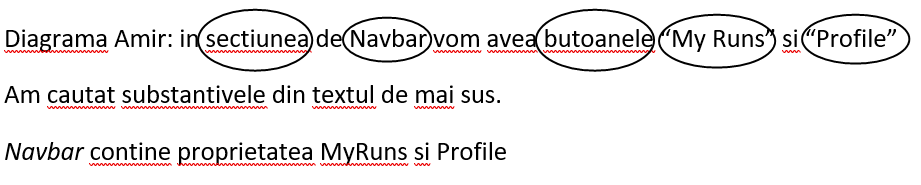
2 entitati

Comportament <------🡪 analiza gramaticala <------🡪 Structura

Usecase: Class:

Cine face? Cum?

Ce face?



**29.06.2023**

UC (Use Case) Diagram / Class Diagram (structura)

Descoperim clase si relatiile dintre ele - interactiune dintre niste obiecte = relatie

Ne ajuta sa vedem cum se intampla un comportament. Interpretarea acelui comportament.

Eg. “display view” (aplicatia va afisa un view care va contine….x, y, z)

Use caseul trebuie sa aiba o legatura cu diagrama de clase (legatura dintre ele)

Determinarea claselor (abstractizare) – Obiecte: proprietati

**04.07.2023**

Class Diagram Relations: Aggregate, Composite, Multiplicity

Composite – legatura stransa intre clase

Daca clasa contine butoanele ex yes and no, este relatie Composite

**24.08.2023**

Clasa are proprietati si functionalitati

Campul username din Clase:

Proprietati: numar maxim caractere, dimensiunea

Functionalitati: afiseaza, memoreaza user name, delete username,

Abstractizare si incapsulare:

**Mostenirea** – este tot o relatie dintre clase

Aplicarea principiului de **mostenire** – se pleaca de la copii, nu de la parinte; ai un set de clase (copii), de unde mai departe se creeaza parintele -> rezulta principiul de **re-use.**

Avem clasa tractor, autobuz, tir, BMW, Suzuki (motocicleta)

Proprietati: motor\_capacitate, diametru\_volan, numar roti, pozitie\_faruri, greutate

Functionalitati: seteaza\_capacitate\_motor, seteaza\_diam\_volan, regleaza\_pozitie\_faruri.

Se creeaza apoi o *clasa de baza* (generica) – **Vehicul**

Proprietati: motor\_capacitatre, diámetro\_volan, numar\_roti, pozitie\_faruri, greutate

Functionalitati: seteaza\_capacitate\_motor, seteaza\_diam\_volan, regleaza\_pozitie\_faruri

Relatia dintre ele va fi o sageata cu cap gol, linie intreaga, iar derivatele clase le vor mosteni de la clasa de baza. Clasa Generica se creeaza dupa ce au fost creeate clasele copii.

*Clasa de baza* (parinte, contine proprietatile si functionalitatile claselor derivate) si *Clasa derivata* (fiu)

Clasele se identifica dupa:

* Identificare **Substantive** – clase candidat
* Relatii Content
* Composite / Agregate
* Multiplicity
* Use Case – metode

**05.09.2023**

1. Class / Metode / Functie (functionalitate)

2. Boundary / Control / Entity class (acestea sunt cele 3 categorii de clase)

Clasa Boundary – rol interactiune (interfata, afisare), preluare, transmite info din interior, din app spre exterior, spre alta app (de ex. Butonul de autentificare, popup, textbox sunt **clase boundary**).

Clasele Boundary trebuie sa aibe *functia* **set ( )** pentru memorare informatie si functia **get ( )** pentru a transmite informatia mai departe.

Clasa Control – rol de gestionare si control al fluxului de informatie (ex. clasa “Baza de data conturi”, sau “Nume Cont”, Parola Cont”, “Adresa Email”).

Clasa Entity – rol de memorare al informatiei in memoria nevolatila, pentru a putea fi folosita de aplicatie (“Cont Utilizator” care are ca si functionalitate “Inregistreaza Cont”.

O **clasa** contine **proprietati** si **metode de functii**

Intre parantezele functiei, se trece logica functionalitatii in cuvinte, care apoi se vor transforma in cod.

IMAGINEAZA-TI LOGICA FUNCTIILOR si CE FACI CU OUTPUTUL LOR (transmit, memorez…)

METODELE CLASELOR TREBUIESC STABILITE

Determinare, sortare (functii) – **mesaje noi primite** – ce face cu outputul functia din aplicatia ta

**07.09.2023**

1. Class / Metode <--> UC

2. Function Logic

**07.19.2023**

1. OOT + (3) - (3)

2. Web Programing (Limbajul de programare Java Script) Impact

- download **Vue js**

***Creeaza clasa “baza de date” si detaliaza in requirements***

***Clase boundry entity control***

***Principiu multiplicitate 1..0 intre clase***

