Structura aplicatiei:

Smartcardul contine :

- Id-ul studentului -> un numar int
- Vector de materii care contine:
    - id –ul materiei -> numar byte
    - Numarul de note -> numar byte
    - Vector de note care contine:
        - Nota -> nr byte(intre 1 si 11)
        - Ziua -> nr byte
        - Luna -> nr byte
        - Anul -> nr short

In aplicatie sunt retinute urmatoarele informatii:

Pinul cardului utilizat -> variabila globala ***globalPin*** (vector de byte)

Id – ul studentului current -> variabila globala ***globalStudentID*** (int)

Variabila de tip Student, ce retine informatii legate de studentul current in variabila globala : ***currentGlobalStudent***,

structura definite In felul urmator:

id – ul studentului -> numar int

vector de materii care contine:

- id –ul materiei -> numar byte
- Numarul de note -> numar byte
- Daca taxa este platita sau nu -> boolean
- Vector de note care contine:
    - Nota -> nr byte(intre 1 si 11)

Ziua -> nr byte

Luna -> nr byte

Anul -> nr short

Functionalitatile smartcardului sunt reprezentate de functiile :

1) Verify -> INS 0x20, primeste comanda: LC -> lungimea pinului, urmata de pin, nu returneaza output, si verifica daca pinul primit este cel correct si seteaza daca este cel corect(pin.isValidated() devine true) .
2) updatePin -> INS 0x70, primeste comanda: LC lungimea pinului current, si lungimea noului pin + 2, urmata de : lungimea pinului curent, pinul current, lungimea noului pin, noul pin, nu returneaza output, si daca pinul vechi introdus este corect, schimba pinul la lungimea si valoarea noului pin(resetand numarul de incercari).
3) updateID -> INS 0x71, primeste comanda: LC valoarea 4 urmata de 4 bytes care reprezinta noul id al studentului, nu returneaza output
4) addGrade -> INS 0x90, primeste comanda: LC valoarea 6 urmata de 6 bytes care reprezinta, in aceasta ordine : id - ul notei, valoarea notei, ziua luna si anul(reprezentat de ultimii 2 bytes) si returneaza 4 bytes, reprezentand id – ul studentului
5) getSpecificGrades -> INS 0x91, primeste comanda: LC numarul de materiilor la care dorim sa aflam ultima nota valida, si LC bytes, reprezentand id – urile materiilor. Returneaza LC bytes, fiecare din ele fiind : valoarea ultimei note intre 1 si 10, sau 0 daca nu exista nota, sau daca nu exista note valide(se ignora nota eroare de 11).
6) getAllGradesOneSubject -> INS 0x93, primeste comanda: LC valoarea 1 urmata de id – ul materiei la care dorim sa aflam toate notele, si returneaza valorile tuturor notelelor existente pe smartcard de la acea materie, in ordinea cronologica dupa data introducerii acestora pe smartcard.
7) updateLastGradeOneSubject -> INS 0x92, primeste comanda : LC valoarea 2 urmata de id – ul materiei la care se modifica ultima nota retinut, si noua nota. Aceasta functie inlocuieste valoarea ultimei note de la acea materie de pe smartcard cu valoarea introdusa, nu returneaza output.
8) getID -> INS 0x98, primeste comanda: LC valoarea 0, si returneaza 4 bytes reprezentand id – ul studentului.

Functiile 4 -> 7 vor functiona doar daca pinul a fost verificat cu success anterior.

Pe card materiile au id – uri numere intregi intre 1 si 5, oride alt id ducand la eroare.

Structura terminalului:

Aplicatia terminal (java) contine 2 componente conceptuale:
a) interactiunea cu baza de date
b) interactiunea cu smartcardul

a) *Interactiunea cu baza de date* este reprezentata de clasa Database din pachetul Ops care contine functiile:

- getMapObjectNames -> citeste fisierul care contine materiile si id – urile lor si intoarce un map care reprezinta idMaterie -> numeMaterie

- getStudentIntel -> primeste un id de student, si citeste fisierul id.txt returnand o structura de tip Student cu informatiile din acel fisier

- printStudentIntel -> suprascrie informatiile din fisierul studentID.txt cu informatiile retinute in variabila ***currentGlobalStudent***

- addGradeToStudentDB -> primeste id – ul materiei si detaliile unei note(valoarea / data calendaristica) si o adauga in variabila ***currentGlobalStudent.***

- payTax -> primeste ID – ul unei materii si plateste taxa acelei materii (seteaza isTaxPayed pe true)

b) *Interactiunea cu smartcardul* este reprezentata de clasa CardOps din pachetul Ops si contine functiile de comunicare cu smartcardul:

- validatePin -> primeste pinul si returneaza daca acesta a fost validat(raspuns 90 00 de la smartcard)
- updatePin -> primeste vechiul si noul pin si trimite comanda de update pin la card, returneaza daca operatia a fost executata cu succes

- updateID -> primeste noul id al studentului current si intoarce daca a fost actualizat cu success
- getID -> returneaza id – ul studentului curent existent pe smartcard
- addGradeReturnID -> primeste id – ul unei materii si informatii despre o nota si le trimite la smartcard, acestea fiind adaugate in smartcard, outputul smatrcardului fiind id – ul studentului, care este mai departe valoarea de return a functiei(o valoare int reprezentand id – ul studentului)
- updateLastGrade -> primeste id – ul unei materii si valoarea notei, si le trimite la smartcard pentru ca acesta sa actualizeze valoarea ultimei note de la acea materie cu valoarea noii note introduce. nu returneaza output
- getSpecificGrades -> primeste un vector care contine id – urile notelor la care se doreste sa se afle ultima nota valida, sau 0 daca nu exista note valide la acea materie. Se returneaza un vector de aceeasi dimensiuni cu notele valide sau 0.
- getAllGradesOneSubject -> primeste id – ul unei materii si returneaza toate notele existente pe smartcard la acea materie.

Structura Bazei de date:

Baza de date este reprezentata de o serie de fisiere text(.txt) care au denumirea : *id – ul studentului* urmata de *".txt"*(exemplu : 2.txt contine informatiile studentului cu id 2) si are urmatoarea structura:

Linia 1 : id – ul studentului

Incepand cu linia 3, avem aceeasi structura pentru cele 5 materii:

Linia 1 : id – ul materiei

Linia 2 : numarul de note

Incepand cu urmatoarea linie, timp de 2 * *numar de Note* linii vor fi informatii legate de notele de la acea materie(I de la 1 la *numar de note*):

Linia 2 * I : valoarea notei

Linia 2 * I + 1 : data notei sub formatul : zz / ll / aaaa

O linie care contine o valoare *true / false* si reprezinta daca a fost platita taxa la acea materie

Intre linia cu id - ul studentului si informatiile obiectelor, respective intre fiecare materie exista o linie libera.

Fisierul ObjectNames.txt reprezinta numele materiilor si Id – urile acestora sub forma : idMaterie numeMaterie

In terminal mai exista clase ajutatoare : in pachetul Structs sunt definite clasele Grade / Student / Subject (nota, student, materie), iar in pachetul Ops sunt definite in clase diverse operatii / variabile ajutatoare : Constants, Globals, Operations.

Cele 3 use case – uri sunt definite in clasa Worflows prin cele 4 functii workflow1(2, 3, 4) :
workflow 1 : primeste id – ul unei materii si nota, si adauga nota la acea materie in baza de date si in smartcard(iar daca apare eroare, se introduce 11 in ambele)

Workflow 2 : primeste id – ul unui student, id – ul materiei si nota si actualizeaza baza de date cu acea nota.

Workflow 3 : primeste id – ul unui student si actualizeaza notele care nu exista pe smartcard cu notele din baza de date

Workflow 4 : primeste id – urile meteriilor la care se doreste sa se afle ultima nota valida, si se terurneaza un HashMap care reprezinta idNota -> valoare nota.

Testare: pe parcursul proiectului, rulat diferite serii de comenzi si combinatii din cele existene pe smartcard si din terminal spre smartcard, iar concret, functia trialRun din clasa Workflows utilizeaza cele 4 wokflow – uri.