Test practic

➢ Test

Acest test este compus dintr-o serie de task-uri, organizate în două secțiuni:
1. Definirea claselor și tipurilor necesare
2. Folosirea acestor clase pentru crearea unei aplicații ce respectă cerințele din acest document

Rezolvarea cerințelor se poate face în orice ordine.

In cadrul aplicației se va încerca simularea unui cont de depozit oferit de banca. Un cont de depozit constă în depunerea unei sume de bani în acesta și scoaterea ei după o anumită perioadă de timp. Cu cât perioada de timp în care suma de bani este lăsată în depozit, cu atât banca va oferi o dobândă mai mare pentru aceasta. Un client ce optează pentru un astfel de cont, va avea de ales momentul în care va putea scoate banii din contul depozit: după 1 an, după 2 ani sau după 3 ani. Astfel, banca va oferi o dobândă de 6%, 8% si respectiv 10%. Banca nu mai oferă dobanda acumulată dacă suma de bani sau o parte din ea este scoasă înainte de termen, **dar permite clientului să adauge bani în contul depozit.**

Se dorește crearea pentru fiecare client a unui fișier de log în care se vor reține:

- sumele depuse periodic în contul depozit
- dobânda acumulată la un moment dat

Puteți crea alte metode sau câmpuri decât cele specificate, dacă vi se par relevante și vă ajută la îndeplinirea task-urilor. De asemenea, tot ce nu este explicit specificat în cerințe este lăsat la latitudinea voastră.

➢ Secțiunea 1 – 80p

Enum – 3p

- Definiți un enum numit **AccountType**, ce va fi folosit pentru specificarea termenului limită după care clientul va putea retrage banii din contul depozit fără a pierde suma de bani provenită din dobândă. Valorile pe care le va putea lua o variabilă de tip **AccountType** vor fi**: One_Year, Two_Years** respectiv **Three_Years.**

Delegat – 5p

- Definiți un delegat cu numele **Deposit** ce va fi folosit pentru crearea unui eveniment
    - Tip: **void**
    - Primește doi parametri **sender**, de tipul **object** și **args** de tipul **DepositArgs** (clasă ce o veți implementa ulterior)

Clasa BankAccount - 14p

- **2p** - Membrii de tip variabilă ai acestei clase:
    - Un string nestatic, protected, **name** ce va reține numele clientului.
    - Un int static, protected, **nrGenerator** ce va fi folosit pentru realizarea unui identificator unic pentru fiecare instanță de tipul BankAccount. Acesta va fi inițializat la 1234.
    - Un int nestatic protected **accountId** ce va reține id-ul clientului, unic pentru fiecare instanță în parte.
    - Un string nestatic protected **logFile** ce va reține numele fișierului de log pentru fiecare abonat în parte.

- **3p** – Clasa va conține un constructor **public**
    - Primește un parametru, un **string** ce va fi folosit pentru inițializarea numelui
    - Tot în constructor veți inițializa câmpul **accountId** la valoarea **nrGenerator**, urmând să îl incrementați pe cel din urmă.

- **4p** – Două metode folosite pentru returnarea câmpurile **name,** si **accountId**, care se vor numi **GetName** și **GetAccountId.**

- **5p –** O metodă de tipul void, numită **Info** ce va primi ca parametru un **string** în care veți salva datele clientului respectiv. **Atenție, această metodă va trebui suprascrisă în clase derivate.**

- Fișierele de log au numele de forma '1249.txt', unde 1249 este codul de abonat.

Clasa DepositAccount - 31p

- Derivează clasa BankAccount și implementează interfața IDisposable.

- **1p** - Membrii de tip variabilă ai acestei clase:
    - Un boolean nestatic privat **withdrawRequest** - un flag care, atunci cand este setat pe **true**, clientul va informa banca de faptul că vrea să retragă banii anticipat (înainte de termen).
    - Un decimal nestatic privat **amount** - pentru a reține suma de bani depusă în contul de tip depozit.
    - Un decimal nestatic privat **interestAmount** – pentru a reține suma de bani provenită din dobânda oferită de bancă. Pentru simplificare, vom considera că banca va oferi dobânda în

momentul depunerii unei sume de bani. Dacă suma de bani va fi retrasă înainte de termen, **interestAmount** nu va mai fi oferit.

- Un câmp de tipul AccountType.

- **1p** – Clasa va conține un constructor public
  - Primește doi parametri, un **string** ce va fi folosit pentru inițializarea numelui, si un **AccountType** pentru inițializarea tipul de cont.

- **3p** – Metoda MakeADeposit

o Primește un parametru – suma de bani pe care clientul vrea să o depună. Se actualizează **amount** și **interestAmount** în funcție de termenul limită pentru care a optat clientul.

1An - 6% dobândă pentru suma depusă.
2Ani – 8% dobândă pentru suma depusă.
3Ani – 10% dobândă pentru suma depusă.

Scrieți în fișierul de log data, suma depusă, suma curentă din cont (**amount**) și suma totală de bani acumulată din dobândă în momentul respectiv (**interestAmount**).

- **1p** – Metoda WithdrawNow
  Este de tipul void și nu primește nici un parametru. Setează câmpul withdrawRequest la valoarea **true**.

- **3p** – Suprascrieți metoda **Info**. La informatiile inițiale adăugați în string suma de bani **(amount)** și suma totală de bani acumulată din dobândă (**interestAmount**).

- **4p** – In fișierul 'evidența.txt' trebuie tinuta evidența tutror instantelor de tipul **DepositAccount**.

Astfel la crearea unui cont depozit si la distrugerea unui cont depozit de tip **DepositAccount**, scrieți un mesaj semnificativ in acest fișier. Hint- folosiți metoda Dispose și constructorul.

Clasa DepositArgs - 7p

- Implementează clasa **EventArgs**.
- **1p** - Membrii de tip variabilă ai acestei clase:
  - Un int nestatic, privat, **accountId**.
  - Un int nestatic, privat, **amount** ce reprezintă suma depusă în cont.
- **2p** – Clasa va conține un constructor public
  - Primește doi parametri, pentru inițializarea celor două câmpuri.

- **4p** – Două proprietăți **read-only** pentru câmpurile **clasei.**

Clasa Bank - 20p

- **2p** - Membrii de tip variabilă ai acestei clase:
  - Un eveniment de tipul **Deposit**, privat numit **depositMoney**. Evenimentul va fi pornit în momentul în care se realizează un depozit.
  - Un ArrayList numit **clients**. In această colecție vom reține conturile de tip **BankAccount**.
- **2p** – O metodă publică, de tipul void, fară parametri, **Init()**, folosită pentru inițializarea listei de clienți.

Adăugați in lista mai multe elemente de tipul **DepositAccount**(cu toate cele trei termene limită din **AccountType**).

- **3p** - Un index de tipul **BankAccount**, ce primește ca parametru un int. Va returna contul corespunzător parametrului.

- **5p** – O metodă **CheckAccount**(object sender, DepositArgs args), de tipul void.

În această metodă veți parcurge lista și dacă întâlniți un cont al cărui **accountId** este egal cu câmpul **accountId** din variabila **args,** atunci dacă **withdrawRequest** este **true**, eliminați contul din listă.

- **5p** – O metodă **DepositMoney**(object sender, DepositArgs args), de tipul void.

În această metodă veți parcurge lista și dacă întâlniți un produs al cărui **accountId** este egal cu câmpul **accountId** din variabila **args** atunci apelați metoda **MakeADeposit** pentru respectivul cont.

Metoda **DepositMoney** și metoda **CheckAccount** reprezintă metodele handler pentru evenimentul **depositMoney**.

- **3p** – O metodă publică **DepositTrigger**(`int accountId, decimal amount`), de tipul void.

Scopul acestei metode este de a declanșa evenimentul.

- **5p** – O metodă publică **List**, de tipul void, fără parametri.

Afisați informații despre toate conturile(accountId, tip, amount) din lista de clienți.

➢ Secțiunea 2 – 20p

În clasa **Program** generată default veți avea metoda **Main()** în care veți realiza urmatoarele operații:

- Creați o instanță a clasei **Bank** și apelați metoda **Init.**
- Realizați un meniu
  - o opțiune pentru listare
  - o opțiune pentru realizarea unui depozit.
- Salvați si compilați programul.
- Verificați fișierele de log.