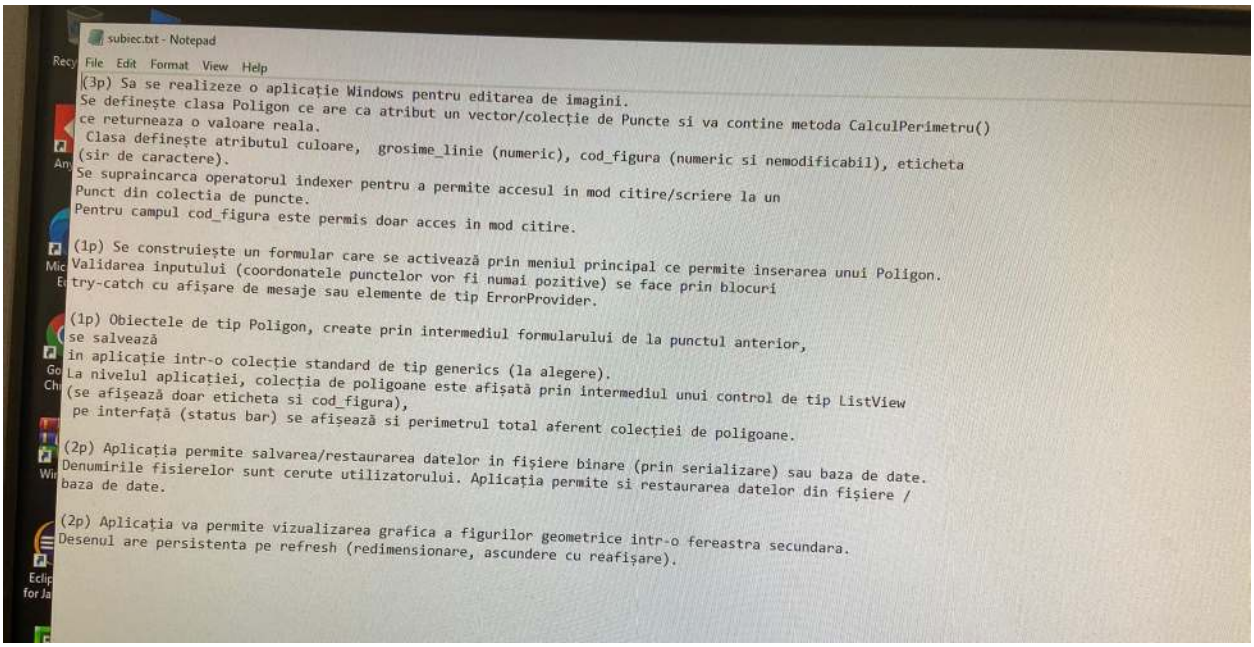
Cam ce se da la PAW

Programarea aplica□iilor Windows (Academia de Studii Economice din Bucure□ti)

Subiecte PAW info ec romana M. D.

Is aproape toate care pot pica cred, bafta!!



(3p) Sa se realizeze o **aplicație Windows, de tip MDI**, pe platforma .NET pentru editarea de imagini. Se vor folosi controale disponibile prin .NET Framework.

Se definește **clasa Punct** cu atributele private: **X** (numeric), **Y** (numeric). Pentru aceasta clasa se definește doar un constructor cu parametri si proprietăți (permit doar citire) pentru atributele private.

Se definește clasa **FiguraGeometrica** ce are ca atribut un vector/colecție de **Puncte**. Se definește **interfața IMasurabil** ce anunța metoda CalculPerimetru() ce returneaza o valoare reala.

Se definește clasa **Triunghi** derivata din **FiguraGeometrica** si din **IMasurabil.** Clasa definește atributul **culoare** (System.Drawing.Color sau structura pentru RGB), **grosime_linie** (numeric), **cod_figura** (numeric si nemodificabil), **eticheta** (sir de caractere). La nivel de clasa se definește numărul fix de puncte, **nr_puncte**, ce definește figura printr-un câmp ce nu își poate modifica valoare si care nu se regăsește in structura obiectelor. Pentru aceasta clasa se definește un constructor cu parametri si se redefinește metoda CalculPerimetru() ce returnează perimetrul triunghiului. Se implementează metoda *Clone()* (definire directa/derivare din interfata *ICloneable*). Se supraincarca operatorul indexer pentru a permite accesul in mod citire/scriere la vectorul **Puncte**. Pentru campul **cod_figura** este permis doar acces in mod citire.

(1p) Se construiește un formular (de tip child) ce permite inserarea unui **Triunghi**. Formularul permite si inserarea valorilor din vectorul **Puncte (**posibilități: parsare text, inserare repetata de valori, generare automata de controale, etc. Numărul acceptat de puncte este preluat din **nr_puncte**). Validarea inputului se face prin blocuri try-catch cu afișare de MessageBox-uri sau elemente de tip ErrorProvider. La nivelul aplicației, formularul se activează prin meniul principal.

(1p) Obiectele de tip **Triunghi,** create prin intermediul formularului de la punctul anterior, se salvează in aplicație intr-o colecție standard de tip generics (la alegere).

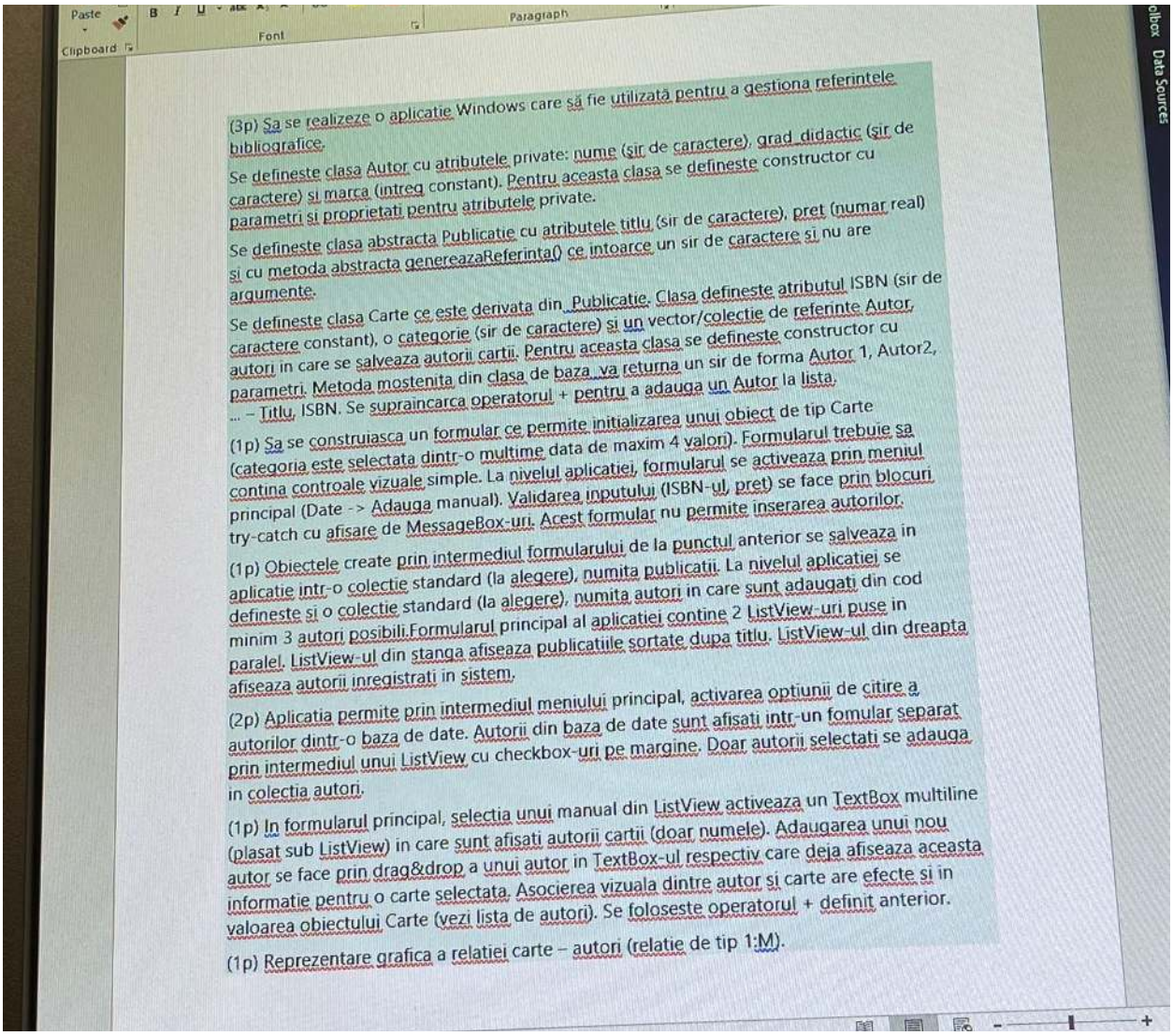
La nivelul aplicației, colecția de figuri este afișată prin intermediul unui control de tip ListView (se afișează doar **eticheta**, **cod_figura** iar elementul are culoarea din Triunghi) intr-un alt formular de tip child. De asemenea, pe interfață (status bar sau toolbar) se afișează si perimetrul total aferent colecției de triunghiuri.

(1p) Aplicația permite salvarea/restaurarea datelor in fișiere binare (prin serializare). Denumirile fisierelor sunt cerute utilizatorului. Aplicația permite restaurarea datelor din fișiere în TextBox-uri multiline la nivelul formularului copil principal.

(2p) Aplicația permite prin intermediul meniului principal, activarea opțiunii de salvare a datelor intr-o baza de date (se recomanda MS Access). Numele bazei de date se cere utilizatorului si se considera ca are structura necesara salvării datelor (numele tabelei se presupune cunoscut si are structura – cod figura | eticheta | perimetru).

(1p) Considerând propriul sistem de coordonate, aplicația permite vizualizarea grafica a figurilor geometrice intr-un alt formular de tip copil. Desenul are persistenta pe refresh (redimensionare, ascundere cu reafișare).

(1p) Se implementează sincronizare si la nivelul formularului cu desen. Printr-un meniu contextual se poate copia in Clipboard o descriere a triunghiului selectat. Meniul contextual mai permite inserarea unui triunghi cu valori default la locația de pe desen dată de cursorul mouse-ului.



(3p) Sa se realizeze o aplicatie Windows care să fie utilizată pentru a gestiona referintele bibliografice.

Se defineste clasa Autor cu atributele private: nume (șir de caractere), grad_didactic (șir de caractere) si marca (intreg constant). Pentru aceasta clasa se defineste constructor cu parametri si proprietati pentru atributele private.

Se defineste clasa abstracta Publicatie cu atributele titlu (sir de caractere), pret (numar real) si cu metoda abstracta genereazaReferinta() ce intoarce un sir de caractere si nu are argumente.

Se defineste clasa Carte ce este derivata din Publicatie. Clasa defineste atributul ISBN (sir de caractere constant), o categorie (sir de caractere) si un vector/colectie de referinte Autor, autori in care se salveaza autorii cartii. Pentru aceasta clasa se defineste constructor cu parametri. Metoda mostenita din clasa de baza va returna un sir de forma Autor 1, Autor2, ... – Titlu, ISBN. Se supraincarca operatorul + pentru a adauga un Autor la lista.

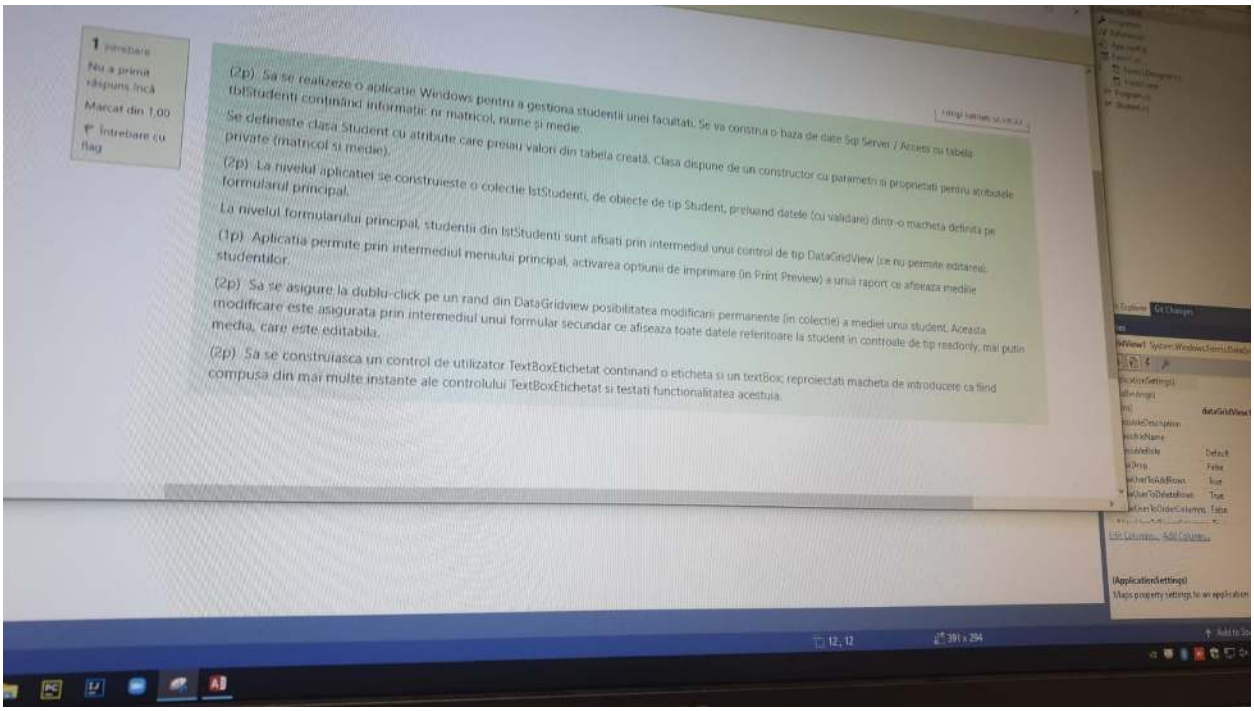
(1p) Sa se construiasca un formular ce permite initializarea unui obiect de tip Carte (categoria este selectata dintr-o multime data de maxim 4 valori). Formularul trebuie sa contina controale vizuale simple. La nivelul aplicatiei, formularul se activeaza prin meniul principal (Date -> Adauga manual). Validarea inputului (ISBN-ul, pret) se face prin blocuri try-catch cu afisare de MessageBox-uri. Acest formular nu permite inserarea autorilor.

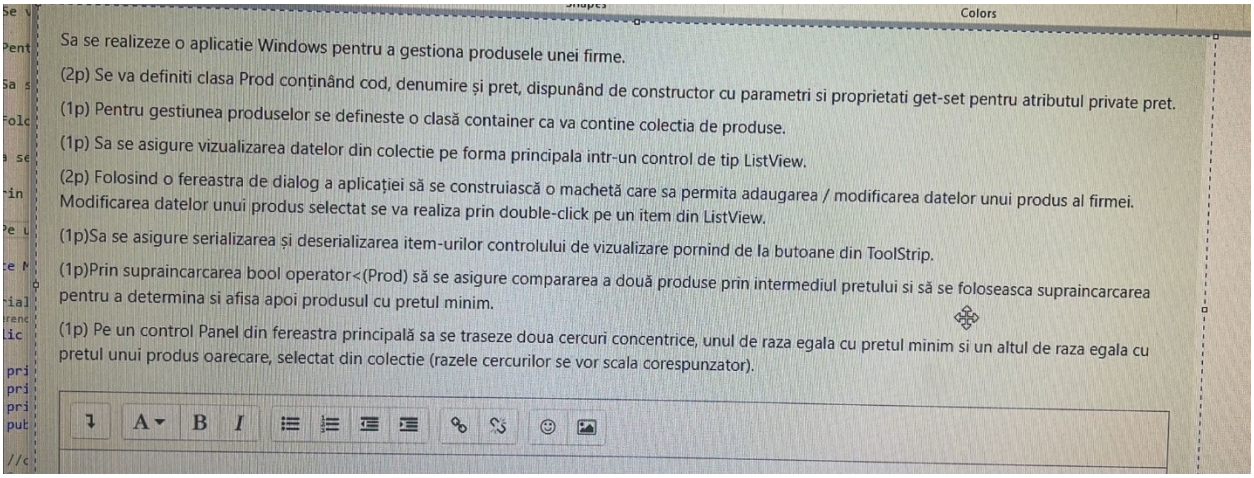
(1p) Obiectele create prin intermediul formularului de la punctul anterior se salveaza in aplicatie intr-o colectie standard (la alegere), numita publicatii. La nivelul aplicatiei se defineste si o colectie standard (la alegere), numita autori in care sunt adaugati din cod minim 3 autori posibili.Formularul principal al aplicatiei contine 2 ListView-uri puse in paralel. ListView-ul din stanga afiseaza publicatiile sortate dupa titlu. ListView-ul din dreapta afiseaza autorii inregistrati in sistem.

(2p) Aplicatia permite prin intermediul meniului principal, activarea optiunii de citire a autorilor dintr-o baza de date. Autorii din baza de date sunt afisati intr-un fomular separat prin intermediul unui ListView cu checkbox-uri pe margine. Doar autorii selectati se adauga in colectia autori.

(1p) In formularul principal, selectia unui manual din ListView activeaza un TextBox multiline (plasat sub ListView) in care sunt afisati autorii cartii (doar numele). Adaugarea unui nou autor se face prin drag&drop a unui autor in TextBox-ul respectiv care deja afiseaza aceasta informatie pentru o carte selectata. Asocierea vizuala dintre autor si carte are efecte si in valoarea obiectului Carte (vezi lista de autori). Se foloseste operatorul + definit anterior.

(1p) Reprezentare grafica a relatiei carte – autori (relatie de tip 1:M).





- Definiți clasa **AccessPackage** cu proprietățile **Id** (int), **Name** (string) și **Price** (double). Definiți clasa **Registration** cu proprietățile **CompanyName** (string), **NoOfPasses** (int) și **AccessPackageId** (int). Clasele vor dispune de constructori cu parametrii.
- Pachetele de acces disponibile vor fi încărcate din fișierul AccessPackages.txt. Fișierul text va fi creat cu ajutorul unui editor de text la alegere și va conține minim 3 intrări.

(2.5p)

- Instanțele clasei **Registration** vor putea fi adăugate prin intermediul unui formular secundar, care va implementa validările necesare. Utilizatorul va putea alege pachetul de acces folosind un control ComboBox. Instanțele clasei vor fi stocate într-o colecțiestandard și vor fi afișate în formularul secundar prin intermediul unui control de tip **DataGridView**.
- Utilizatorul va avea posibilitatea de a modifica (dublu click) sau de a șterge înregistrările.

(1p)

- Implementați operatorul de cast explicit la **double** pentru a calcula costul fiecărei înregistrări. Afișați costul total pentru toate înregistrările utilizând un control **StatusStrip** și actualizați-l pentru orice modificare a listei de înregistrări.

(1p)

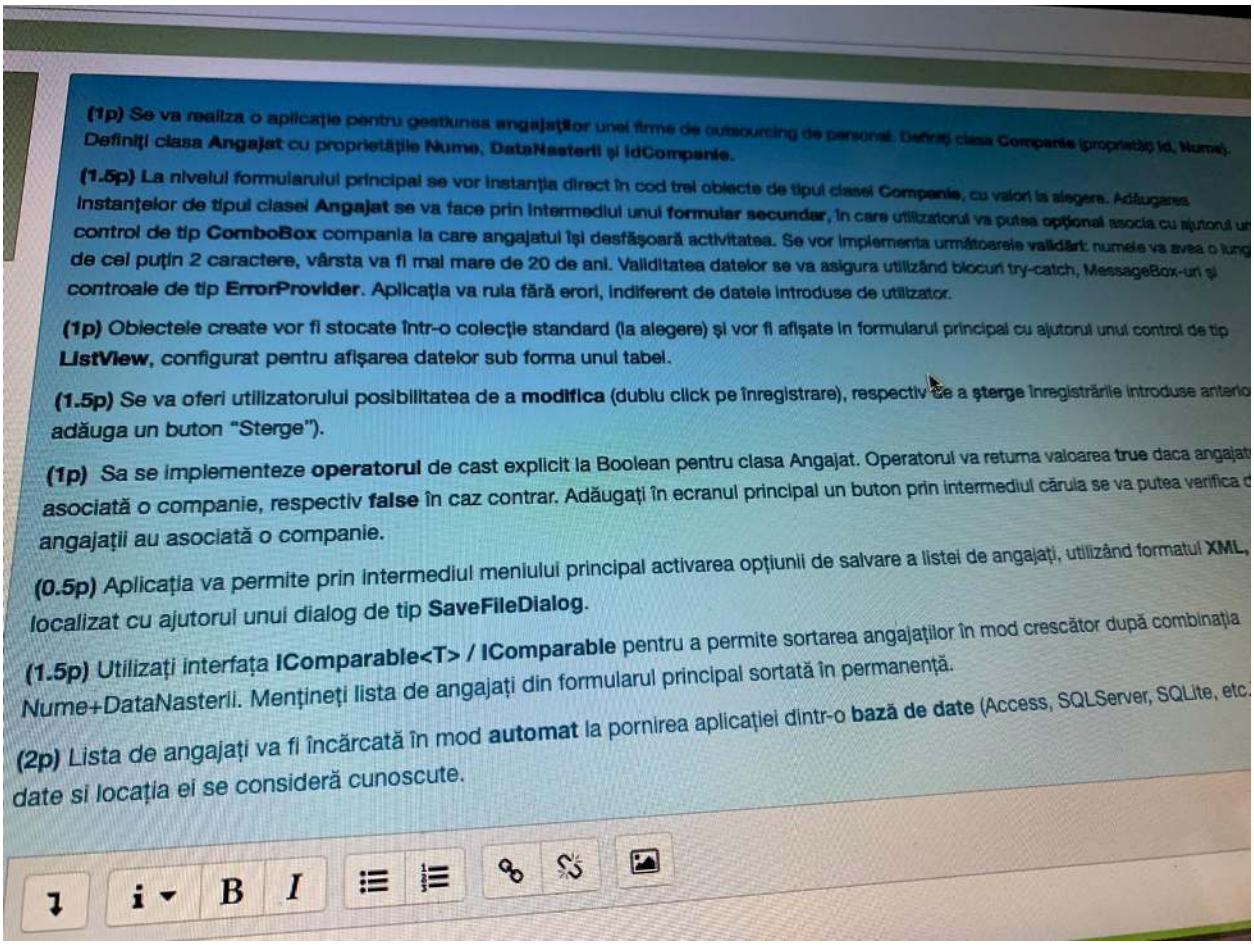
- Aplicația va persista automat lista înregistrărilor într-un fișier binar. Lista înregistrărilor va fi restaurată automat din fișierul binar la pornirea aplicației (dacă există fișierul).

(2p)

- Implementați interfața **IComparer <T> / IComparer** pentru a permite utilizatorului să sorteze lista înregistrărilor în ordine ascendentă în funcție de numele companiei, respectiv în funcție de tipul de pachet de acces. Utilizatorul va putea alege ordinea de sortare din formularul principal.

(1p)

- Aplicația permite tipărirea (print preview) a unei **diagrame** care conține numărul de înregistrări pentru fiecare tip de pachet de acces.



(1p) Se va realiza o aplicație pentru gestiunea **angajaților** unei firme de outsourcing de personal. Definiți clasa **Companie** (proprietăți **Id, Nume**). Definiți clasa **Angajat** cu proprietățile **Nume, DataNasterii** și **IdCompanie**.

(1.5p) La nivelul formularului principal se vor instanția direct în cod trei obiecte de tipul clasei **Companie**, cu valori la alegere. Adăugarea instanțelor de tipul clasei **Angajat** se va face prin intermediul unui **formular secundar**, în care utilizatorul va putea **opțional** asocia cu ajutorul un control de tip **ComboBox** compania la care angajatul își desfășoară activitatea. Se vor implementa următoarele **validări**: numele va avea o lung de cel puțin 2 caractere, vârsta va fi mai mare de 20 de ani. Validitatea datelor se va asigura utilizând blocuri try-catch, MessageBox-uri și controale de tip **ErrorProvider**. Aplicația va rula fără erori, indiferent de datele introduse de utilizator.

(1p) Obiectele create vor fi stocate într-o colecție standard (la alegere) și vor fi afișate în formularul principal cu ajutorul unui control de tip **ListView**, configurat pentru afișarea datelor sub forma unui tabel.

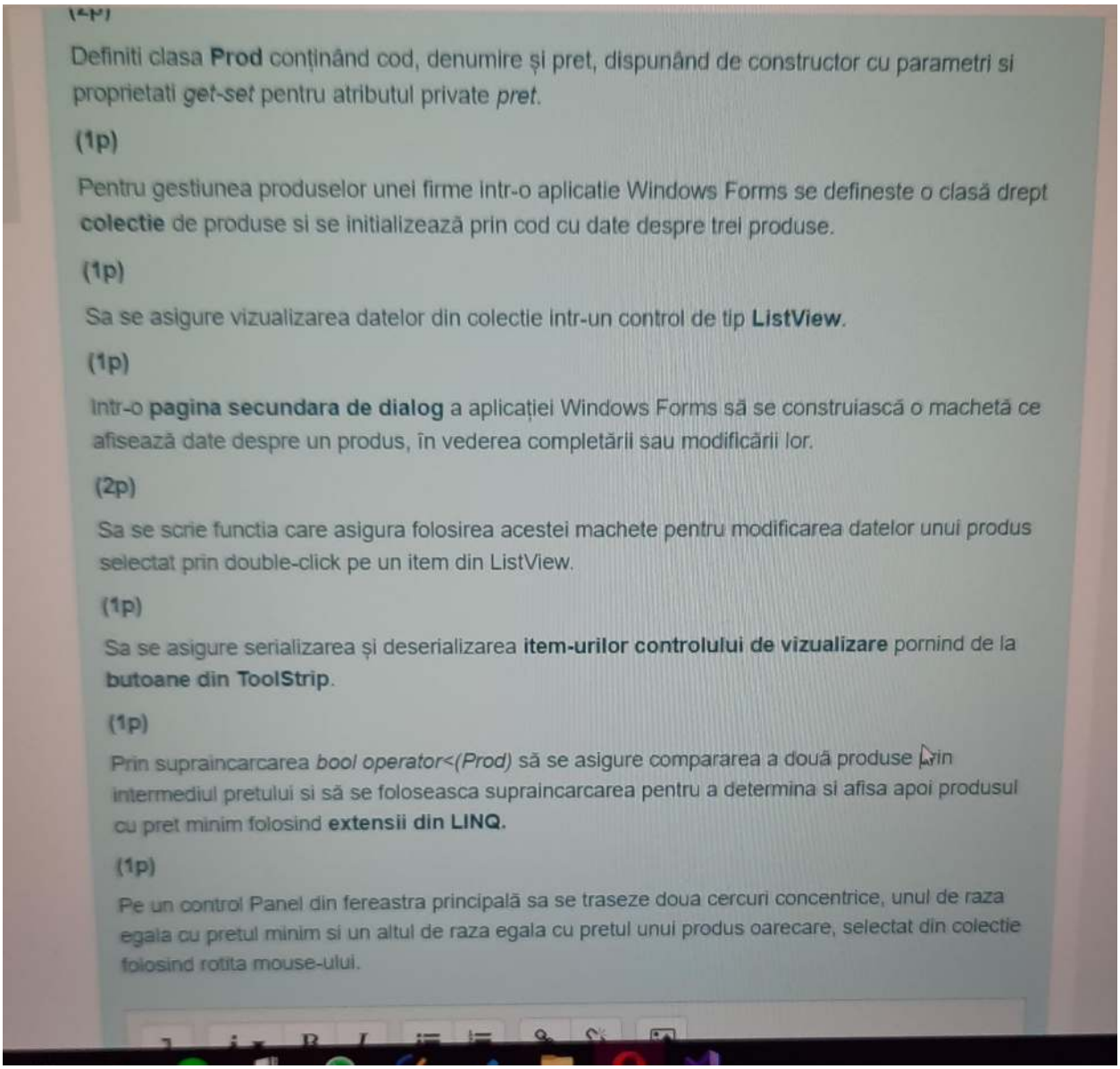
(1.5p) Se va oferi utilizatorului posibilitatea de a **modifica** (dublu click pe înregistrare), respectiv de a **șterge** înregistrările introduse anterio adăuga un buton "Sterge").

(1p) Sa se implementeze **operatorul** de cast explicit la Boolean pentru clasa Angajat. Operatorul va returna valoarea **true** daca angajat asociată o companie, respectiv **false** în caz contrar. Adăugați în ecranul principal un buton prin intermediul căruia se va putea verifica c angajații au asociată o companie.

(0.5p) Aplicația va permite prin intermediul meniului principal activarea opțiunii de salvare a listei de angajați, utilizând formatul **XML**, localizat cu ajutorul unui dialog de tip **SaveFileDialog**.

(1.5p) Utilizați interfața **IComparable<T> / IComparable** pentru a permite sortarea angajaților în mod crescător după combinația Nume+DataNasterii. Mențineți lista de angajați din formularul principal sortată în permanență.

(2p) Lista de angajați va fi încărcată în mod **automat** la pornirea aplicației dintr-o **bază de date** (Access, SQLServer, SQLite, etc. date si locația ei se consideră cunoscute.



(2p) *Sa se realizeze o aplicatie Windows C# .NET pentru a gestiona studentii unei facultati. Se va construi o baza de date Sql Server / Access cu tabela* **tblStudenti** *conținând informații despre matricol, nume și medie.*

Se defineste **clasa Student** *cu atribute care preiau valori din tabela creată. Clasa dispune de un constructor cu parametri si proprietati pentru atributele private (matricol si medie).*

(2p) *La nivelul aplicatiei se construieste o colectie* **lstStudenti**, *de obiecte de tip* **Student**, *preluand datele (cu validare) dintr-o macheta definita pe* **formularul principal**.

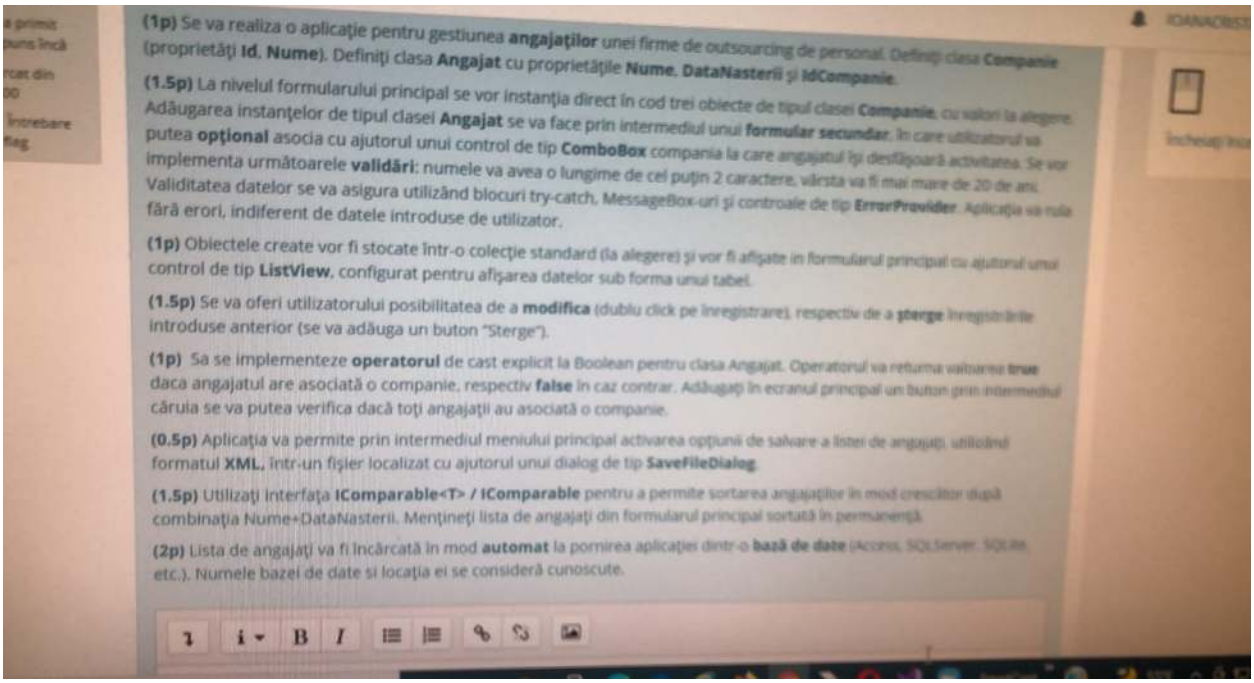
La nivelul formularului principal, studentii din **lstStudenti** *sunt afisati prin intermediul unui control de tip DataGridView (ce nu permite editarea).*

(1p) *Aplicatia permite prin intermediul meniului principal, activarea optiunii de imprimare (in* **Print Preview**) *a unui raport ce afiseaza mediile studentilor.*

(2p) *Sa se asigure la dublu-click pe un rand din DataGridview posibilitatea modificarii permanente (in colectie) a mediei unui student. Aceasta modificare este asigurata prin intermediul unui formular secundar ce afiseaza toate datele referitoare la student in controale de tip readonly, mai putin media, care este editabila.*

(2p) *O optiune a meniului principal permite salvarea datelor din colectie in tabela* **tblStudenti**.

(1p) *Sa se construiasca un control de utilizator* **TextBoxEtichetat** *continand o eticheta si un textBox; reproiectati macheta de introducere ca fiind compusa din mai multe instante ale controlului* **TextBoxEtichetat** *si testati functionalitatea acestuia.*



(1p) Se va realiza o aplicaţie pentru gestiunea **angajaţilor** unei firme de outsourcing de personal. Definiţi clasa **Companie** (proprietăţi **Id**, **Nume**). Definiţi clasa **Angajat** cu proprietăţile **Nume**, **DataNasterii** şi **IdCompanie**.

(1.5p) La nivelul formularului principal se vor instanţia direct în cod trei obiecte de tipul clasei **Companie**, cu valori la alegere. Adăugarea instanţelor de tipul clasei **Angajat** se va face prin intermediul unui **formular secundar**. În care utilizatorul va putea **opţional** asocia cu ajutorul unui control de tip **ComboBox** compania la care angajatul îşi desfăşoară activitatea. Se vor implementa următoarele **validări**: numele va avea o lungime de cel puţin 2 caractere, vârsta va fi mai mare de 20 de ani. Validitatea datelor se va asigura utilizând blocuri try-catch, MessageBox-uri şi controale de tip **ErrorProvider**. Aplicaţia va rula fără erori, indiferent de datele introduse de utilizator.

(1p) Obiectele create vor fi stocate într-o colecţie standard (la alegere) şi vor fi afişate în formularul principal cu ajutorul unui control de tip **ListView**, configurat pentru afişarea datelor sub forma unui tabel.

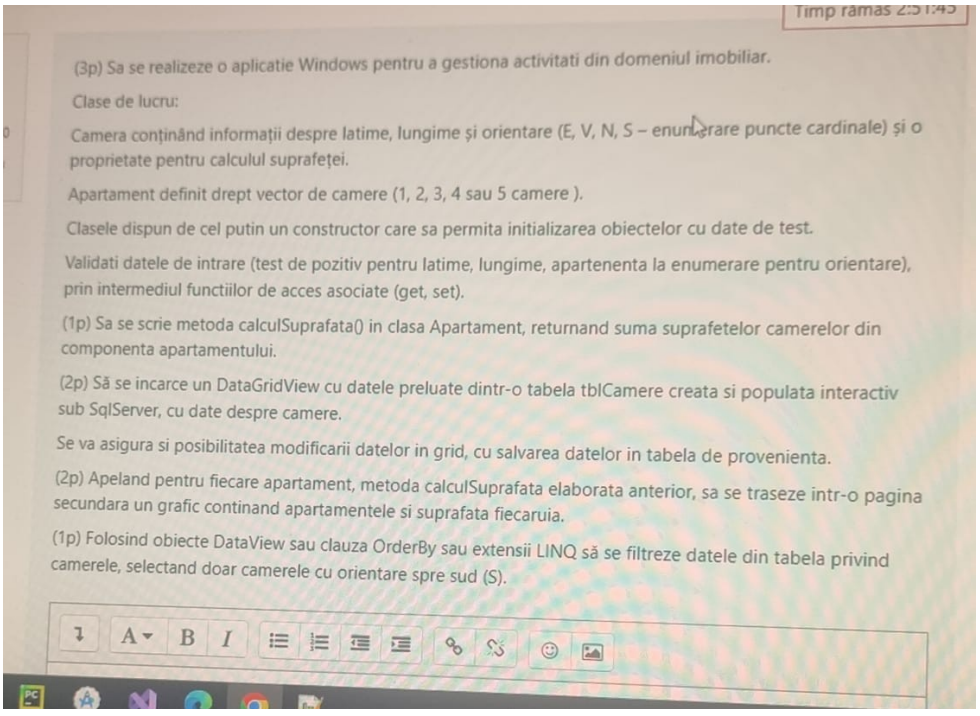
(1.5p) Se va oferi utilizatorului posibilitatea de a **modifica** (dublu click pe înregistrare), respectiv de a **şterge** înregistrările introduse anterior (se va adăuga un buton "Sterge").

(1p) Să se implementeze **operatorul** de cast explicit la Boolean pentru clasa Angajat. Operatorul va returna valoarea **true** daca angajatul are asociată o companie, respectiv **false** în caz contrar. Adăugaţi în ecranul principal un buton prin intermediul căruia se va putea verifica dacă toţi angajaţii au asociată o companie.

(0.5p) Aplicaţia va permite prin intermediul meniului principal activarea opţiunii de salvare a listei de angajaţi, utilizând formatul **XML**, într-un fişier localizat cu ajutorul unui dialog de tip **SaveFileDialog**.

(1.5p) Utilizaţi interfaţa **IComparable<T> / IComparable** pentru a permite sortarea angajaţilor în mod crescător după combinaţia Nume+DataNasterii. Menţineţi lista de angajaţi din formularul principal sortată în permanenţă.

(2p) Lista de angajaţi va fi încărcată în mod **automat** la pornirea aplicaţiei dintr-o **bază de date** (Access, SQLServer, SQLite, etc.). Numele bazei de date şi locaţia ei se consideră cunoscute.



Timp ramas 2:51:45

(3p) Sa se realizeze o aplicatie Windows pentru a gestiona activitati din domeniul imobiliar.

Clase de lucru:

Camera conţinând informaţii despre latime, lungime şi orientare (E, V, N, S – enumerare puncte cardinale) şi o proprietate pentru calculul suprafeţei.

Apartament definit drept vector de camere (1, 2, 3, 4 sau 5 camere).

Clasele dispun de cel putin un constructor care sa permita initializarea obiectelor cu date de test.

Validati datele de intrare (test de pozitiv pentru latime, lungime, apartenenta la enumerare pentru orientare), prin intermediul functiilor de acces asociate (get, set).

(1p) Sa se scrie metoda calculSuprafata() in clasa Apartament, returnand suma suprafetelor camerelor din componenta apartamentului.

(2p) Să se incarce un DataGridView cu datele preluate dintr-o tabela tblCamere creata si populata interactiv sub SqlServer, cu date despre camere.

Se va asigura si posibilitatea modificarii datelor in grid, cu salvarea datelor in tabela de provenienta.

(2p) Apeland pentru fiecare apartament, metoda calculSuprafata elaborata anterior, sa se traseze intr-o pagina secundara un grafic continand apartamentele si suprafata fiecaruia.

(1p) Folosind obiecte DataView sau clauza OrderBy sau extensii LINQ să se filtreze datele din tabela privind camerele, selectand doar camerele cu orientare spre sud (S).

(1p) Să se realizeze o aplicație Windows pentru gestionarea creditelor acordate de către o bancă. Clasa Credit contine: client, valoare credit, dobanda, data acordarii, perioada pentru care se acorda. Un formular secundar activat din bara de instrumente a aplicației permite citirea datelor pentru un obiect de tip Credit.

(1p) Fereastra principală a aplicației afișează lista de credite într-un control de tip DataGridView și utilizează formularul creat anterior pentru adăugarea de elemente noi. Obiectele create sunt stocate într-o colecție standard. Pentru testarea rapidă, aplicația va inițializa lista de obiecte de tip Credit la pornire, prin citirea valorilor dintr-un fișier text.

(2p) Aplicația permite selecția unui Credit din listă și modificarea acestuia prin intermediul formularului construit la primul punct. Se va implementa validarea datelor și afișarea erorilor prin intermediul unui obiect ErrorProvider.

(3p) Se va defini delegatul AlgoritmDesfasurator care va putea referi funcții care au ca parametru un obiect de tip Credit și returnează un vector de obiecte decimal, reprezentând valoarea de plată din fiecare lună.

Se definește clasa Algoritmi care va conține două metode de calcul ce respectă prototipul delegatului și implementează următoarele modalități de calcul pentru desfășurător:

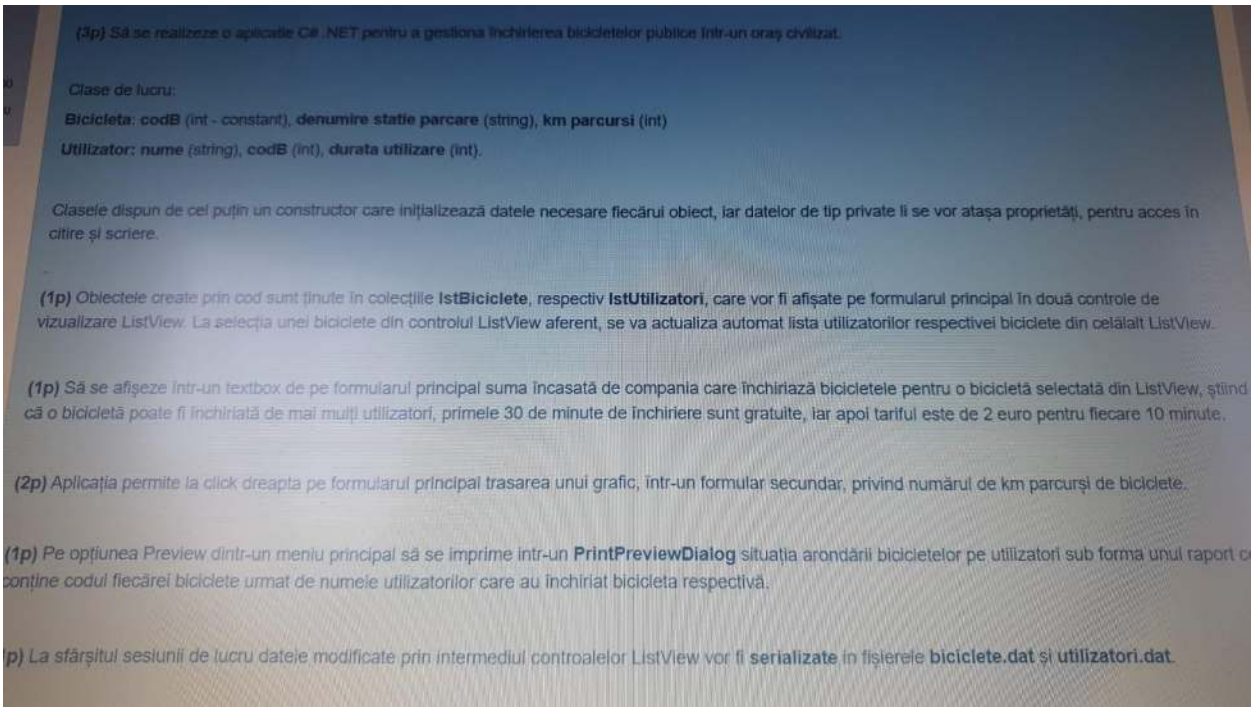
· DesfasuratorRateDescrescătoare: valoarea creditului se împarte la numărul de luni; în fiecare lună se va plăti valoarea calculată anterior și dobânda la suma rămasă de plată;

· DesfasuratorRateConstante: se calculează dobânda totală de plată; valoarea de plată pentru fiecare lună se obține împărțind valoarea creditului adunată cu dobânda totală, la numărul de luni.

Se va completa clasa Credit pentru a gestiona o referință la un delegat AlgoritmDesfasurator și se va adăuga metoda publică CalculDesfășurător care să apeleze metoda referită de către obiectul delegat.

În formularul de introducere / modificare se adaugă un control de tip ComboBox pentru selectarea algoritmului de calcul pentru desfășurător.

(2p) Implementați suportul pentru a permite utilizarea mecanismului de Clipboard pentru transferul de credite între două instanțe ale aplicației.



(3p) Să se realizeze o aplicație C# .NET pentru a gestiona închirierea bicicletelor publice într-un oraș civilizat.

Clase de lucru:

Bicicleta: **codB** (int - constant), **denumire statie parcare** (string), **km parcursi** (int)

Utilizator: nume (string), **codB** (int), **durata utilizare** (int).

Clasele dispun de cel puțin un constructor care inițializează datele necesare fiecărui obiect, iar datelor de tip private li se vor atașa proprietăți, pentru acces în citire și scriere.

(1p) Obiectele create prin cod sunt ținute în colecțiile **lstBiciclete**, respectiv **lstUtilizatori**, care vor fi afișate pe formularul principal în două controle de vizualizare ListView. La selecția unei biciclete din controlul ListView aferent, se va actualiza automat lista utilizatorilor respectivei biciclete din celălalt ListView.

(1p) Să se afișeze într-un textbox de pe formularul principal suma încasată de compania care închiriază bicicletele pentru o bicicletă selectată din ListView, știind că o bicicletă poate fi închiriată de mai mulți utilizatori, primele 30 de minute de închiriere sunt gratuite, iar apoi tariful este de 2 euro pentru fiecare 10 minute.

(2p) Aplicația permite la click dreapta pe formularul principal trasarea unui grafic, într-un formular secundar, privind numărul de km parcurși de biciclete.

(1p) Pe opțiunea Preview dintr-un meniu principal să se imprime într-un **PrintPreviewDialog** situația arondării bicicletelor pe utilizatori sub forma unui raport c conține codul fiecărei biciclete urmat de numele utilizatorilor care au închiriat bicicleta respectivă.

p) La sfârșitul sesiunii de lucru datele modificate prin intermediul controalelor ListView vor fi **serializate** în fișierele **biciclete.dat** și **utilizatori.dat**.

(3p) Sa se realizeze **aplicație Windows, de tip MDI,** care să fie utilizată pentru a gestiona comenzile unei pizzerii.

Se implementează **clasa Topping** pentru care se definesc atributele:

- denumire – sir de caractere | **preț** – numeric real | **cantitate** – numeric real | **cod** – numeric întreg nemodificabil

Se definește interfața **ICustomizabil** ce anunța metoda CalculCostPizza().

Se implementează clasa **ComandaPizza** ce este derivata din **ICustomizabil** si **ICloneable**. Se definesc atributele:

- **nume** – sir de caractere | **blat** – sir de caractere | **durataRealizare** – numeric întreg;

· **topping** – vector/colecție de elemente de tip **Topping**

Pentru clasa **ComandaPizza** se definesc metodele:

· proprietăți ce permit citirea pentru **nume**, **durata** si **topping** (indexer)

· DOAR constructor cu parametri;

· se supraîncarcă > si < pentru a compara 2 obiecte de tip **ComandaPizza** in funcție de costul total al pizzei (de folosește metoda CalculCostPizza())

Metoda CalculCostPizza() va calcula costul total al produsului pe baza datelor din **topping**. Costul de baza pentru orice tip de pizza, indiferent de topic, este definit printr-o valoare statica si constanta.

(1p) Sa se construiască un formular ce permite inițializarea unui obiect de tip **ComandaPizza** (tipul de blat este selectat dintr-o mulțime data de maxim 4 valori). Formularul trebuie sa contina controale vizuale simple. La nivelul aplicației formularul se activează prin meniul principal. Validarea inputului se face prin blocuri try-catch cu afișare de MessageBox-uri si controale de validare.

(1p) Obiectele create prin intermediul formularului de la punctul anterior se salvează in aplicație intr-o **colecție generics** la alegere .

generics la alegere .

La nivelul formularului principal, fisele sunt afișate prin intermediul unui control de tip **TreeView** fiind grupate după tipul blatului (mulțime de maxim 4 valori definita si utilizata la punctul anterior).

(1p) Aplicația permite citirea datelor dintr-un fișier de tip text (XML sau fișier in care valorile atributelor sunt scrise pe linie) ce conține datele pentru o pizza, fara topping, doar numele, tipul de blat si durata preparării.

(2p) Aplicația permite prin intermediul meniului principal, activarea opțiunii de citire a opțiunilor de topping dintr-o baza de date. Numele bazei de date si locația ei se cer utilizatorului. Topping-ul este afișat intr-un formular separat prin intermediul unui ListView cu checkbox-uri pe margine.

(1p) Acest formular este utilizat si pentru a adaugă topping la vectorul/colecția dintr-o **ComandaPizza**. (Scenariu posibil: selectez o pizza in controlul-tree si printr-un buton activez formularul cu topping. Fac selecția elementelor dorite si dau salvare. Elementele selectate sunt introduse in vectorul **topping** al comenzii). Adăugarea unei tip de topping la **ComandaPizza** se face prin supraîncărcarea operatorului + in clasa.

(1p) Se construiește un grafic care descrie vizual gradul de utilizare al unui tip de topping (aceasta se alege din formular). Desenul are persistenta pe refresh (redimensionare, ascundere cu reafișare).

Sa se realizeze o aplicatie Windows Forms C# .NET pentru a gestiona cosul de cumparaturi al unui client.

Se vor defini doua clase:

- una elementara, care stocheaza datele aferente unui item (produs) si implementeaza metode specifice;
- o clasa pentru a modela multimea de produse din cosul de cumparaturi (**vector**).

(1p) Clasa elementara dispune de cel putin un constructor, o supradefinire a metodei ToString() si o **proprietate** ce calculeaza valoarea unui item, ca produs intre cantitate si pret; campurile private dispun de accesori get/set.

(1p) Clasa care gestioneaza multimea de obiecte dispune de un constructor si de cel putin o supraincarcare a unui operator pentru adaugarea de noi obiecte.

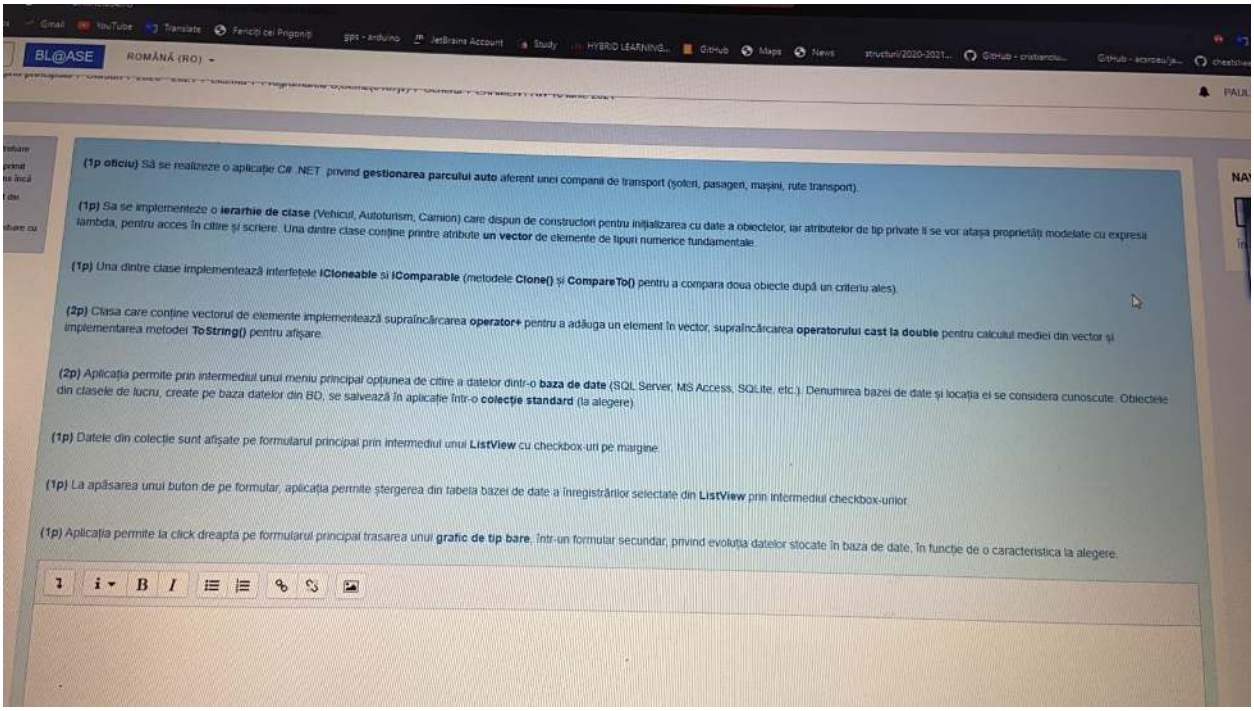
(2p) Prin alegerea unei optiuni din meniul de pe formularul principal se deschide un **formular secundar** care foloseste controale vizuale simple (textBox, numericUpDown, comboBox etc.) pentru introducerea datelor necesare crearii unui item. In functie de **DialogResult**, se preiau datele din macheta si se asigura vizualizarea lor intr-un ListView pe **pagina principala**.

(1p) Obiectele create prin intermediul formularului secundar se salvează în vectorul gestionat de a doua clasa a aplicatiei.

(2p) Obiectele afisate prin intermediul controlului **ListView** pot fi **editate** și **șterse** (inclusiv din vector) cu ajutorul unui meniu contextual.

(2p) La apăsarea unui buton din **toolStrip**-ul de pe formularul principal, aplicația permite **previzualizarea** unui raport sau tabel (folosind **PrintPreviewDialog**), cu produsele din cos (cantitati, pret, valoare) si suma totala de plata.

(1p) La sfârșitul sesiunii de lucru, datele din **controlul de vizualizare** vor fi salvate intr-un fisier pe disc.



Sa se realizeze o aplicatie Windows Forms C# .NET pentru a facilita desfasurarea de interviuri pentru ocuparea unor job-uri.

Se vor defini doua clase:

- una elementara, care stocheaza datele unui interviu și implementeaza interfața **ICalculPunctaj**;
- o clasa pentru a modela multimea de **interviuri asociate unui job** (colectie sau vector).

(1p) Clasa elementara este serializabila si dispune de cel putin un constructor si o supradefinire a metodei **ToString();** campurile private dispun de accesori get/set.

(1p) Clasa care gestioneaza multimea de obiecte dispune de un constructor si de cel putin o supraincarcare a unui operator pentru adaugarea de noi obiecte.

(2p) Prin alegerea unei optiuni din meniul de pe formularul principal se deschide un **formular secundar** care foloseste controale vizuale simple (textBox, calendar, comboBox etc.) pentru introducerea datelor necesare creării unui interviu. La preluarea datelor din macheta se asigura vizualizare lor intr-un ListView pe pagina principala.

(1p) Obiectele create prin intermediul formularului secundar se salvează în colecția (vectorul) gestionata de a doua clasa a aplicatiei, acre dispune și de **un indexer** pentru a adresa un element din multime.

(2p) Obiectele afisate prin intermediul controlului **ListView** pot fi șterse (inclusiv din colecție sau vector) cu ajutorul unui meniu contextual.

(2p) La apăsarea unui buton de pe formularul principal, aplicația permite trasarea unui **grafic de tip bare**, într-un panel pe formularul principal, privind o caracteristica numerica a interviurilor din ListView.

(1p) La sfârșitul sesiunii de lucru, datele din controlul de vizualizare vor fi salvate intr-un fisier pe disc.

Să se implementeze o aplicație Windows pentru înscrierea candidaților la programele de masterat ale unei facultăți.

(1p) Să se implementeze clasa **ProgramStudiu** cu următoarele atribute private:

codProgram (int), **denumireProgram** (string), **numărLocuriBuget** (int), **numărLocuriTaxă** (int).

Se consideră clasa **Candidat** cu următoarele atribute private:

codCandidat (int), **numeCandidat** (string), **medieConcurs** (float), **vectorOptiuni** (int [ ] – vector de coduri programe la care s-a înscris candidatul).

(2p) Să se construiască un formular ce permite inițializarea și afișarea de obiecte din clasele **ProgramStudiu** și **Candidat**. Lista de candidați este afișată într-un control **ListView**, iar programele de studiu într-un control **ListBox**. Atunci când este selectat un candidat din **ListView** și se face click pe un program de studiu din **ListBox**, respectivul program este adăugat în vectorul de opțiuni ale candidatului.

Obiectele **Candidat** create prin intermediul formularului principal se salvează în aplicație într-o colecție standard (la alegere).

(2p) Prin apăsarea unui buton de pe formularul principal se deschide un formular secundar care conține controale vizuale simple pentru introducerea datelor, necesare creării unui nou candidat. Validarea inputului se face prin blocuri try-catch cu afisare de MessageBox-uri si controale de validare. Pentru vectorul de coduri programe la care s-a înscris candidatul se va folosi un TextBox în care codurile sunt separate prin virgulă.

(1p) Obiectele **Candidat** afișate prin intermediul controlului **ListView** pot fi editate sau șterse (inclusiv din colecție) cu ajutorul unui meniu contextual având două opțiuni (editare și ștergere a elementului selectat). În cazul editării unui candidat, se va lansa formularul secundar de la punctul anterior, în care controalele sunt precompletate cu datele existente ale candidatului.

(2p) La apăsarea unui buton de pe formularul principal, aplicația permite trasarea unui **grafic de tip bare**, într-un panel de pe un formular secundar, privind numărul de candidați înscriși la fiecare program de studiu.

(1p) Pe opțiunea Preview dintr-un meniu principal să se afișeze cu ajutorul unui **PrintPreviewDialog** situația înscrierilor candidaților sub forma unui raport ce conține codul fiecărui program, urmat de numele candidaților care s-au înscris la programul respectiv.