1/1\_1/5.6

1/1\_2/6.7

1/2\_1/7.8

1/2\_2/4.3

1/3\_1/5.9

1/3\_2/10

1/4\_1/8.9

1/4\_2/7.4

1/5\_1/5.7

1/5\_2/5.1

2/1\_1/6.8

2/3\_2/9.5

2/5\_2/5.9

3/1\_1/3.5

3/2\_2/4.9

3/3\_2/6

3/5\_1/5.7

3/5\_2/2

4/1\_1/10

4/1\_2/2

4/2\_2/3

4/3\_2/4.7

4/5\_2/5.1

5/1\_1/7.9

5/2\_1/9.9

5/3\_1/8.3

5/4\_1/8.2

5/5\_1/5.9

6/1\_1/5.1

6/3\_1/7.1

6/3\_2/9.6

7/1\_1/7.9

8/1\_1/5.4

8/1\_2/8.7

import os

import sys

sys.path.append(os.getcwd() + "\\")

from domain.entities import student, laborator, notare

from termcolor import colored

class note\_repository:

    def \_\_init\_\_(self):

        self.\_\_note = []

    def cautare(self, n):

        """

        Cauta n

        :param n: notarea de cautat

        :type n: notare

        :return: notarea cautata daca exista in lista, None altfel

        :rtype: notare

        """

        for notare in self.\_\_note:

            if n == notare:

                return notare

        return None

    def cautare\_student(self,stud):

        """

        Cauta studentul stud in lista de asignari.

        :param stud: studentul de cautat

        :type stud: student

        :return: studentul cautat daca exista in lista, lista goala altfel

        :rtype: list

        """

        return  [el for el in self.\_\_note if el.getstudent() == stud]

    def cautare\_stud\_lab(self,stud,lab):

        """

        Cauta laboratorul lab in lista de asignari.

        :param lab: laboratorul de cautat

        :type lab: student

        :return: laboratorul cautat daca exista in lista, lista goala altfel

        :rtype: list

        """

        for notare in self.\_\_note:

            if notare.getstudent() == stud and notare.getlaborator() == lab:

                return notare

        return None

    def cautare\_nota(self,nota):

        """

        Cauta nota nota in lista de asignari

        :param nota: nota de cautat

        :type nota: float

        :return: nota cautata daca exista in lista, lista goala altfel

        :rtype: list

        """

        return  [el for el in self.\_\_note if el.getnota() == nota]

    def add(self, notare):

        """

        Adauga un element la lista de asignari

        :param nota: asignarea de adaugat

        :type nota: notare

        :return: -; se adauga nota la lista de note

        :raises: ValueError daca nota a fost deja acordata

        """

        n = self.cautare\_stud\_lab(notare.getstudent(),notare.getlaborator())

        errors = []

        if n is not None:

            errors.append('Nota fost deja acordata.')

        else:

            self.\_\_note.append(notare)

        if len(errors) > 0:

            errors\_string = '\n'.join(errors)

            raise ValueError(errors\_string)

    def get\_all(self):

        """

        Returneaza o lista cu toate notarile facute.

        :rtype: lista de obiecte de tip notare

        """

        return self.\_\_note

    def show\_all(self):

        """

        Afiseaza lista de notari

        :rtype: -; afiseaza lista

        """

        print(colored('Lista de note este: ','cyan'))

        for el in self.\_\_note:

            print(el)

class note\_repository\_file(note\_repository):

    def \_\_init\_\_(self,filename):

        note\_repository.\_\_init\_\_(self)

        self.\_\_filename = filename

        self.\_\_load\_from\_file()

    def \_\_load\_from\_file(self):

        f = open(self.\_\_filename, 'r')

        lines = f.readlines()

        for line in lines:

            student1,labotor1,nota = [token.strip() for token in line.split('/')]

            nota = notare(student1,labotor1,float(nota))

            note\_repository.add(self, nota)

        f.close()

    def \_\_save\_to\_file(self):

        lista = note\_repository.get\_all(self)

        with open(self.\_\_filename, 'w') as f:

            for el in lista:

                string = str(el.getstudent().getstudentID()) + '/' + str(el.getlaborator().getClientId()) + '/' + str(el.getnota()) + '\n'

                f.write(string)

    def add(self, notare):

        """

        Adauga un element la lista de asignari

        :param nota: asignarea de adaugat

        :type nota: notare

        :return: -; se adauga nota la lista de note

        :raises: ValueError daca nota a fost deja acordata

        """

        note\_repository.add(self, notare)

        self.\_\_save\_to\_file()

    def cautare(self, n):

        """

        Cauta n

        :param n: notarea de cautat

        :type n: notare

        :return: notarea cautata daca exista in lista, None altfel

        :rtype: notare

        """

        return note\_repository.cautare(self,n)

    def cautare\_student(self,stud):

        """

        Cauta studentul stud in lista de asignari.

        :param stud: studentul de cautat

        :type stud: student

        :return: studentul cautat daca exista in lista, lista goala altfel

        :rtype: list

        """

        return  note\_repository.cautare\_student(self,stud)

    def cautare\_stud\_lab(self,stud,lab):

        """

        Cauta laboratorul lab in lista de asignari.

        :param lab: laboratorul de cautat

        :type lab: student

        :return: laboratorul cautat daca exista in lista, lista goala altfel

        :rtype: list

        """

        return note\_repository.cautare\_stud\_lab(self,stud,lab)

    def cautare\_nota(self,nota):

        """

        Cauta nota nota in lista de asignari

        :param nota: nota de cautat

        :type nota: float

        :return: nota cautata daca exista in lista, lista goala altfel

        :rtype: list

        """

        return note\_repository.cautare\_nota(self,nota)

    def get\_all(self):

        """

        Returneaza o lista cu toate notarile facute.

        :rtype: lista de obiecte de tip notare

        """

        return note\_repository.get\_all(self)

    def show\_all(self):

        """

        Afiseaza lista de notari

        :rtype: -; afiseaza lista

        """

        note\_repository.show\_all(self)

def test\_add():

    repo\_test = note\_repository()

    student1 = student('1','Alina','211')

    laborator1 = laborator('1\_1','asta','1.1.2021')

    nota = 6

    notare1 = notare(student1,laborator1,nota)

    assert(repo\_test.get\_all() == [])

    repo\_test.add(notare1)

    assert(len(repo\_test.get\_all()) == 1)

    nota = 7

    notare2 = notare(student1,laborator1,nota)

    try:

        repo\_test.add(notare2)

        assert False

    except ValueError:

        assert True

def test\_cautare():

    repo\_test = note\_repository()

    student1 = student('1','Alina','211')

    laborator1 = laborator('1\_1','asta','1.1.2021')

    nota = 6

    notare1 = notare(student1,laborator1,nota)

    repo\_test.add(notare1)

    assert(repo\_test.cautare(notare1) == notare1)

    student2 = student('2','Alina','211')

    laborator2 = laborator('1\_2','asta','1.1.2021')

    nota = 6

    notare2 = notare(student2,laborator2,nota)

    assert(repo\_test.cautare(notare2) == None)

def test\_cautare\_student():

    repo\_test = note\_repository()

    student1 = student('1','Alina','211')

    laborator1 = laborator('1\_1','asta','1.1.2021')

    nota = 6

    notare1 = notare(student1,laborator1,nota)

    repo\_test.add(notare1)

    assert(len(repo\_test.cautare\_student(student1)) == 1)

    student2 = student('2','Alina','211')

    laborator2 = laborator('1\_2','asta','1.1.2021')

    nota = 7

    notare2 = notare(student2,laborator2,nota)

    assert(repo\_test.cautare\_student(student2) == [])

def test\_cautare\_nota():

    repo\_test = note\_repository()

    student1 = student('1','Alina','211')

    laborator1 = laborator('1\_1','asta','1.1.2021')

    nota1 = 6

    notare1 = notare(student1,laborator1,nota1)

    repo\_test.add(notare1)

    assert(len(repo\_test.cautare\_nota(nota1)) == 1)

    student2 = student('2','Alina','211')

    laborator2 = laborator('1\_2','asta','1.1.2021')

    nota2 = 7

    notare2 = notare(student2,laborator2,nota2)

    assert(repo\_test.cautare\_nota(nota2) == [])

def test\_cautare\_stud\_lab():

    repo\_test = note\_repository()

    student1 = student('1','Alina','211')

    laborator1 = laborator('1\_1','asta','1.1.2021')

    nota1 = 6

    notare1 = notare(student1,laborator1,nota1)

    repo\_test.add(notare1)

    assert(repo\_test.cautare\_stud\_lab(student1,laborator1) == notare1)

    student2 = student('2','Alina','211')

    laborator2 = laborator('1\_2','asta','1.1.2021')

    nota2 = 7

    notare2 = notare(student2,laborator2,nota2)

    assert(repo\_test.cautare\_stud\_lab(student2,laborator2) == None)

test\_add()

test\_cautare()

test\_cautare\_student()

test\_cautare\_nota()

test\_cautare\_stud\_lab()

1/aa/1

2/bb/2

3/cc/3

4/dd/4

5/ee/5

6/ff/6

7/gg/7

8/hh/8

9/ii/9

10/jj/10

11/ww/11