SEMINAR 1 SDA

1) pseudocod / convenții : TElement -> tip general/abstract al
2) TAD colecție a <- b atribuire elementelor
   - definire a = b verificare egalitate
   - specificare TComparabil
3) TAD iterator
4) exemple reprezentare
5) implementare python

TAD colecție

-> conține elem de tip TElement (*)
-> elementele se pot repeta
-> nu implică gestiune poziții
-> nu avem garanție asupra ordinii elementelor

TAD colecție

① specificarea domeniului

$C = \{c \mid c$ colecție conform (*), c conține elem de tip TElem$\}$

② specificarea interfeței (operațiile)

1) adaugă (C, e)

2) căutare (C, e)

3) șterge (C, e)

4) creează (C)

5) distruge (C)

6) dimensiune (C)

7) iterator (C, i)

nrAparitii (C, e)

modifică (C, e, eNou)

ordonează (C)

creează (C)

pre: adevărat

post: $c \in C$, c este colecția vidă

adaugă (C, e)

pre: $c \in C$, e: Telement

post: $c' \in C$, $c' = c \cup \{e\}$

caută (C, e)

pre: $c \in C$, e: Telement

post: caută = { adevărat dacă $e \in C$
fals altfel

sterge (C, e)

pre: $c \in C$, e: Telement

post: $c' \in C$, $c' = C \setminus \{e\}$ (OBS: se șterge doar o apariție a lui e, de obicei prima)

dim (C)

pre: $c \in C$

post: dim = nr. total de elemente

distrugere (C)

pre: $c \in C$

post: colecția c a fost distrusă

iterator (C, i)

pre: $c \in C$

post: $i \in I$, i este iterator peste colecția C și reține un prim elem din acesta

TAD ITERATOR

- domeniu

$\mathcal{I} = \{ i \mid i$ iterator peste colecție $c \in C \}$

- interfață

creează (i, c)

următor (i)

element (i, e)

valid (i)

c: Colecție

iterator (c, it)

cât timp valid (it) execută

element (it, e)

tipărește (it, e)

următor (it)

creează (i, c)

pre: $c \in C$

post: $i \in \mathcal{I}$, i iterator peste colecția c și reține un prim elem din această

următor (i)

pre: $i \in \mathcal{I}$

post: $i' \in \mathcal{I}$, i' reține următorul element față de cel reținut de i

element(i, e)

pre: $i \in [1, n]$

post: e : TElement

post: e = elementul referit în mod curent de i

valid(i)

pre: $i \in [1, n]$

post: { adevărat dacă elem. referit de i este valid
{ fals altfel

④ REPREZENTĂRI

1	4	2	5	6	1	1	2

elem 1, 4, 2, 5, 6, 1, 1, 2

```
class Colectie:
    def __init__(self):
        self.__elements = []
...
class Iterator:
    def __creare__(self, c):
        self.__c = c
```

```
    self.__curentIndex = 0
def element(self):
    return self.__c.__ColectieIterator__elements[self.__curentIndex]
def urmator(self):
    self.__curentIndex += 1
def valid(self):
    return self.__curentIndex < self.__c.dim()
```

PerecheElement Frecv

② | (1,3) | (4,1) | (2,2) | (5,1) | (3,1) | (6,1) |

adauga(c, e)

1) e ∈ c => frecv + 1

2) e ∉ c => adaugare pereche noua (e, 1)

sterge(c, e)

1) frecv e > 1 => frecv - 1

2) frecv e = 1 => sterge (e, 1)

```
class Colectie:
def __init__(self):
    self.__pairs = []
```

```
...
class YIterator:
    def __init__(self, c):
        self.__c = c
        self.__currYIndex = 0
        self.__prevCurrent = 0
```