EÖTVÖS LORÁND
UNIVERSITY | BUDAPEST

# Implementing Custom Data Types

Object Oriented Programming | 2024 Spring
Practice 3

Presented by Tarlan Ahadli
Supervised by Prof. Teréz Anna Várkonyi

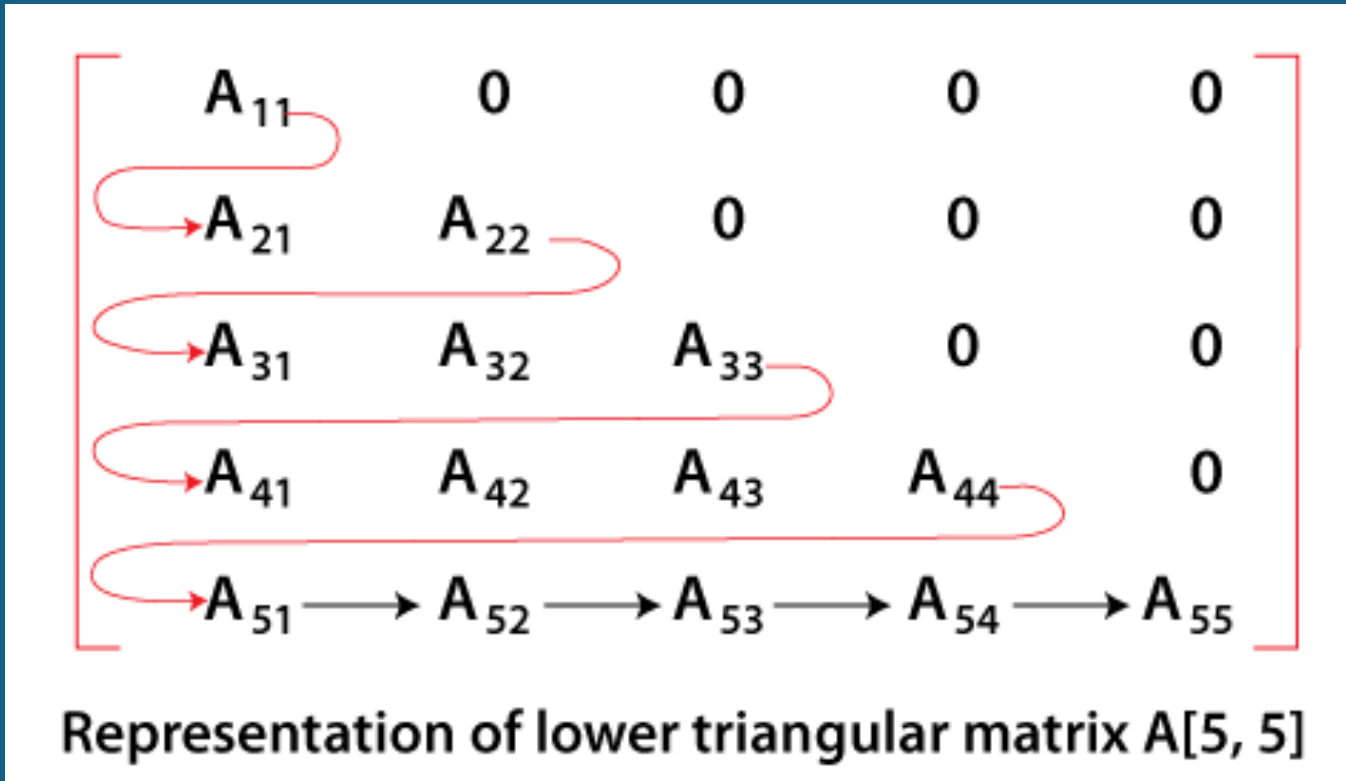
# Lower triangular matrix

- A lower triangular matrix is a type of square matrix where all the elements above the main diagonal are zero.
- Problem: Storing all zeros are useless

$$L = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}.$$

# Lower triangular matrix | Row Major

- A lower triangular matrix in row major ordering
  - $A[5,2] = V[1 + 2 + 3 + 4 + 2]$
  - $A[i,j] = V[\frac{i*(i-1)}{2} + j]$
    - $1 \leq j \leq i \leq N$



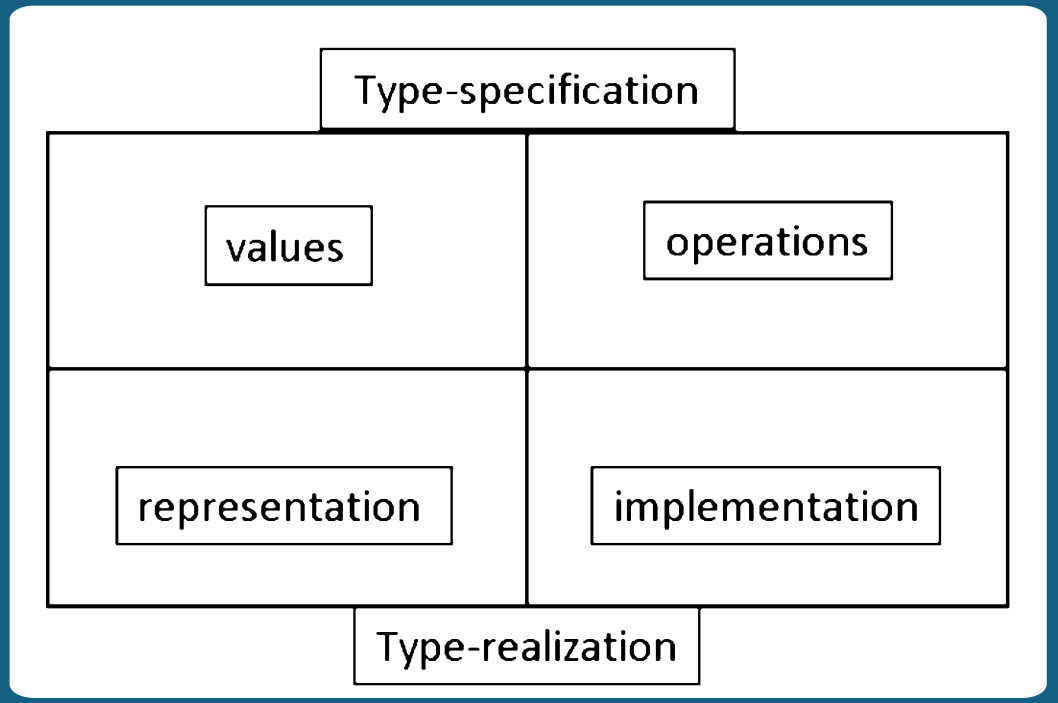

Representation of lower triangular matrix A[5, 5]

# Index conversion function | ind

- Let's define a function given to integers it will convert them row-major integer index
  - $ind: N\ x\ N \rightarrow N$
- C#

```
int ind(int i, int j){
    return (i*(i-1) / 2 + j);
}
```

# Type Definition Table



# Values/Operations

- Values
  - $LTM\ (R^{NxN})$
- Operations
  - GetElement
    - $A = (a{:}\ LTM(R^{NxN}), i, j{:}\ [1 - N], e{:}\ R)$
    - $e\ \coloneqq a[i, j]$
  - SetElement
    - $A = (a{:}\ LTM(R^{NxN}), i, j{:}\ [1 - N], e{:}\ R)$
    - $a[i, j] \coloneqq e$

# Values/Operations | Cont.

- Operations
  - Add
    - $A = \left(a, b: LTM(R^{NxN}), c: LTM(R^{NxN})\right)$
    - $c = a + b$
  - Product
    - $A = \left(a, b: LTM(R^{NxN}), c: LTM(R^{NxN})\right)$
    - $c = a * b$

# Representation/Implementation

- $Representation$
  - $V: R^{\frac{n*(n+1)}{2}}$
- Implementation
  - GetElement
    - $if\ j \leq i$
      - $e := a.v[ind(i,j)]$
    - $else$
      - $e \coloneqq 0$

# Representation/Implementation

- Implementation
  - SetElement
    - $if\ j \leq i$
      - $a.v[ind(i,j)] = e$
    - $else$
      - Throw an exception

# Representation/Implementation

- Implementation
  - Add
    - $\forall\, i \in [1, \ldots, |a.v|], c.v[i] = a.v[i] + b.v[i]$
  - Product
    - $\forall\, i, j \in [1, \ldots, |a.v|]\,, c.v[ind(i,j)] = \sum_{k=1}^{n} a.v[ind(i,k)] * b.v[ind(k,j)]$

# Representation/Implementation

- Implementation
  - Add
    - $\forall\, i \in [1, \dots, |a.v|], c.v[i] = a.v[i] + b.v[i]$
  - Product
    - $\forall\, i, j \in [1, \dots, |a.v|]\,, c.v[ind(i,j)] = \sum_{k=1}^{n} a.v[ind(i,k)] * b.v[ind(k,j)]$
      - Any Idea to improve the algorithm above?

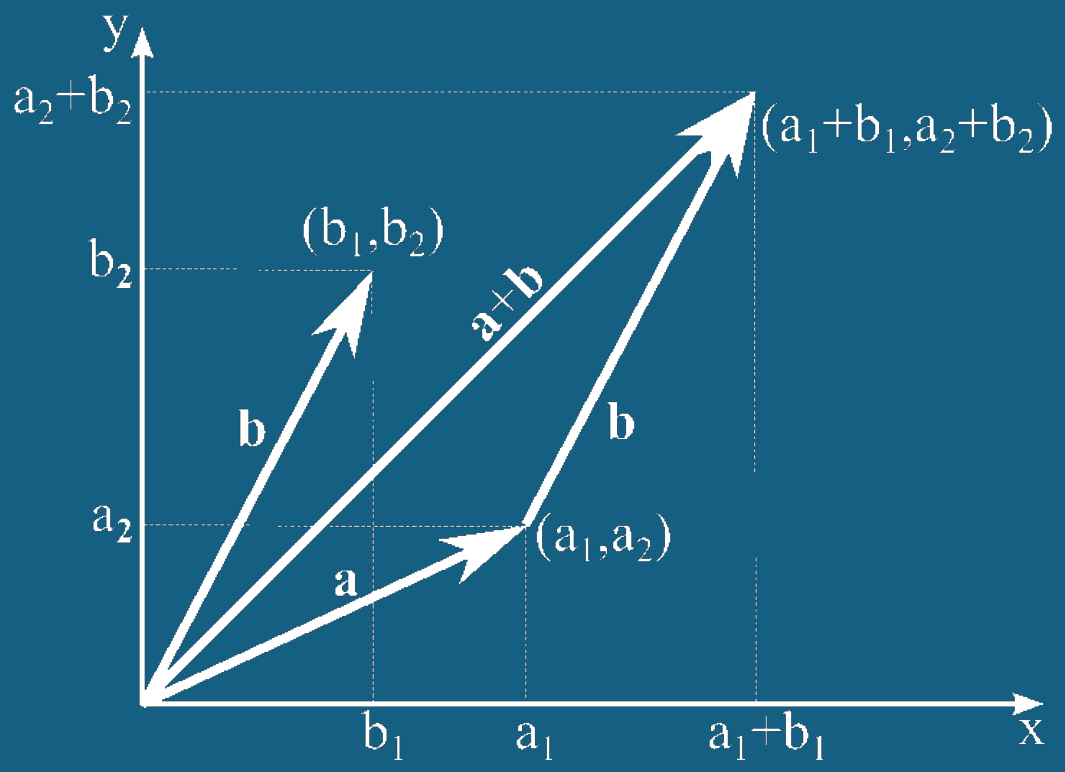
# Representation/Implementation

- Implementation
  - Add
    - $\forall\, i\ \in [1, \dots, |a.v|], c.v[i] = a.v[i] + b.v[i]$
  - Product
    - $\forall\, i, j \in [1, \dots, |a.v|]\,, c.v[ind(i,j)] = \sum_{k=j}^{i} a.v[ind(i,k)] * b.v[ind(k,j)]$
    - $if\ j \leq i$

# Class Diagram

- LTM
- ----------
- -v: double []
- -n: int (matrix dimension)
  - n>0
- ----------
- $+getElement(i: int, j: int): double\ \{query\}$
- $+setElement(i: int, j: int, e: double)\{void\}$
- $\underline{+\ operator + (a: LTM, b: LTM): LTM\ \ \rightarrow Underline\ means\ static}$
- $\underline{+\ operator * (a: LTM, b: LTM): LTM}$
- $-ind(i: int, j: int): int\ \{query\}$

# Data Type: Planar Vectors



# Type Definition Table | Planar Vector

- Value
  - V
- Operations
  - Add
    - $A = (a, b{:}V, c{:}V)$
    - $c = a + b$
  - Stretch
    - $A = (a{:}V, k{:}R, c{:}V)$
    - $c = k * a$
  - *Dot Product*
    - $A = (a, b{:}V, s{:}R)$
    - R= $a * b$

# Type Definition Table | Planar Vector

- Representation
  - $x, y : R$
- Implementation
  - Add
    - $c.x, c.y = a.x + b.x, a.y + b.y$
  - Stretch
    - $c.x, c.y = k * a.x, k * b.y$
  - Dot Product
    - $s = a.x * b.x + a.y * b.y$

# Class Diagram | Planar Vector

- Vector
- ----------
- $-x: double$
- $-y: double$
- ----------
- $+operator + (a: Vector, b: Vector): Vector$
- $+operator * (a: Vector, k: double): Vector$
- $+operator * (a: Vector, b: Vector): double$

# Problem

- Given some vectors is the sum of these vectors normal to another vector?
- Summation
- $A = (a{:}\, V^n, v{:}\, V, l{:}\, L)$
- $Pre = (a = a' \ \wedge \ v = v')$
- $Post = (Pre \wedge l = (v \ * \underbrace{\sum_{i=1}^{n} a_i}_{\text{Summation algorithm pattern}} = 0))$

# Problem | Cont.

- Analogy
- $m \sim 1$
- $(H, +, 0) \sim (V, +, \mathbf{0})$
- $f(i) \sim a[i]$

- Algorithm
- ----------
- $s \coloneqq (0,0)$
- $i = 1 \ldots n$
  - $s \coloneqq s + a[i]$
- $l \coloneqq (s * v = 0)$