Data Types

Object Oriented Programming

2024 Spring

Presented by Tarlan Ahadli
Supervised by Prof. Teréz Anna Várkonyi

Data Types in C#

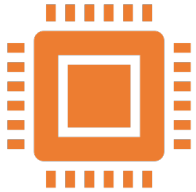
Value Data Types | They contain actual data

Predefined Data Types

- Signed integral: sbyte, short, int, long
- Unsigned integral: byte, ushort, uint, ulong
- Unicode characters: char
- IEEE binary floating point: float, double
- High precision decimal floating point: decimal
- Boolean: bool

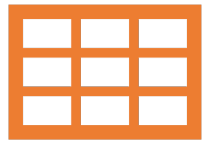
User-Defined Data Types

- enum
- struct



Pointer Data Types | They hold the memory address of another variable

(they are not typically used in safe C# code. Not managed by the .NET runtime's garbage collector*.)*



Reference Data Types | They store a reference to the actual data

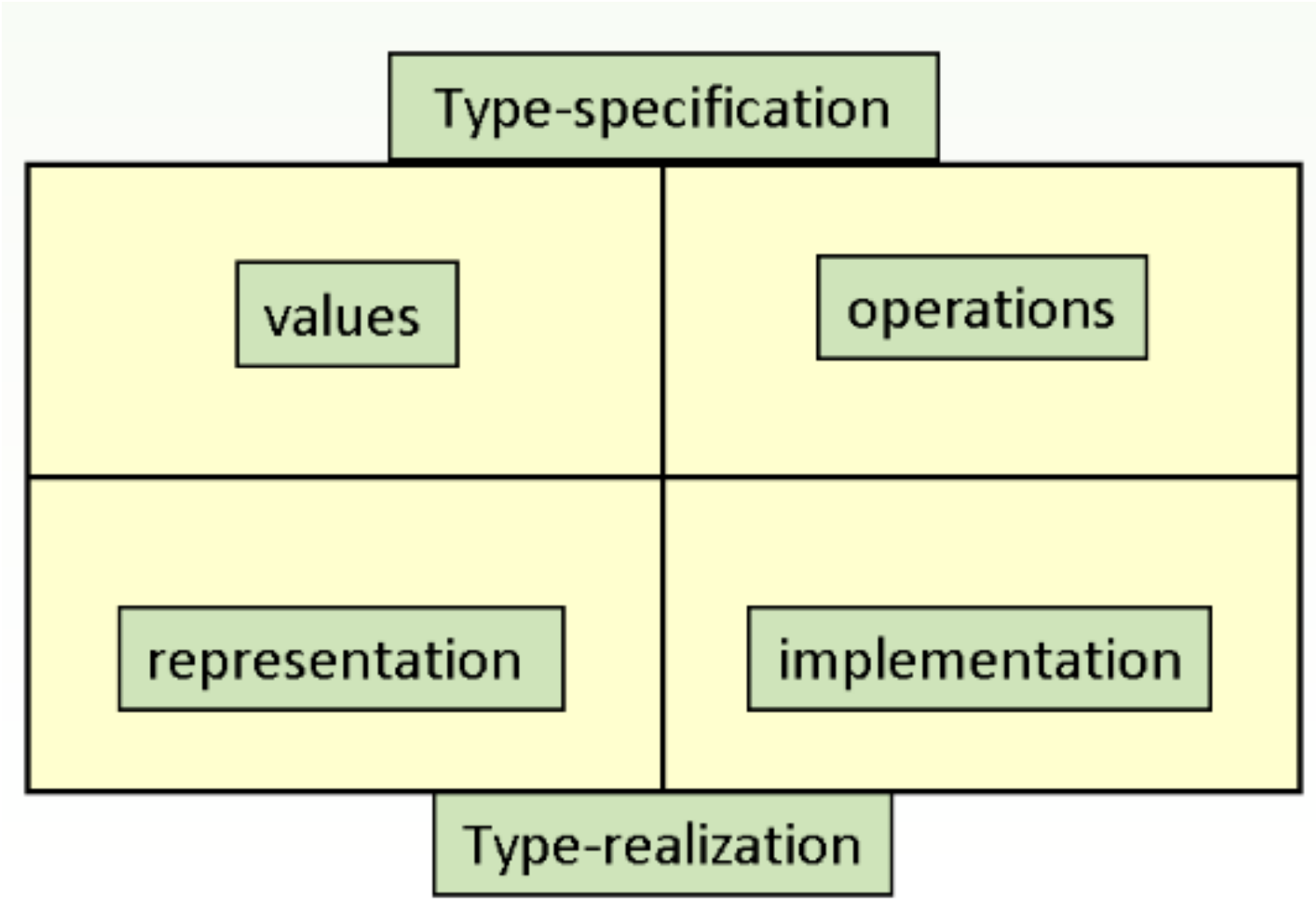
Predefined Data Types

- object
- string
- dynamic

User-Defined Data Types

- class
- interface
- delegate

User Defined Data Types Classes



Type Definition

- Type Specification
  - Values | Range of values we can store
  - Operations | Operations that we can do with this data type
- Type Realization
  - Representation | What values we need to represent the type in the memory
  - Implementation | How operations are carried out

Example: Rational Number Type | C# Code

- Specification
  - Value | Q
  - Operations | given a: Q, b:Q | +, -, *, / | Result c: Q
- Realization
  - Representation | x,y which are integers | y != 0 (x/y)
  - Implementation
    - * | $c.x, c.y := a.x * b.x, a.y * b.y$
    - +| $c.x, c.y =$ a.x * b.y + b.x * a.y, a.y * b.y
    - - | $c.x, c.y =$ a.x * b.y - b.x * a.y, a.y * b.y
    - / | $c.x, c.y =$ a.x * b.y, a.y * b.x

Example: Rational Number Type | C# Code

- Value/Representation

```
class Q {
    public int x;
    public int y;

    public Q(int x, int y){
        if (y == 0)
                throw new ArgumentException("Denominator cannot be zero.", nameof(y));
        this.x = x;
        this.y = y;
    }
}
```

Example: Rational Number Type | C# Code

- Operations/Implementation

```
class Q {
    public static Q operator +(Q a, Q b) {return new Q(a.x * b.y + b.x * a.y, a.y * b.y);}

    public static Q operator -(Q a, Q b){return new Q(a.x * b.y - b.x * a.y, a.y * b.y);}

    public static Q operator *(Q a, Q b){return new Q(a.x * b.x, a.y * b.y);}

    public static Q operator /(Q a, Q b){
    if (b.x == 0)
    throw new DivideByZeroException("Cannot divide by a rational number with a numerator of zero.");

    return new Q(a.x * b.y, a.y * b.x);
    }
}
```

Creating Type Definition Table for Q

Value	Operations
Q	A = (a:Q, b:Q, c:Q) c = a + b
	A = (a:Q, b:Q, c:Q) c = a - b
	A = (a:Q, b:Q, c:Q) c = a * b
	A = (a:Q, b:Q, c:Q) c = a / b | b!=0
x, y ∈ Z | y != 0	| + | c.x, c.y = a.x * b.y + b.x * a.y, a.y * b.y | - | c.x, c.y = a.x * b.y - b.x * a.y, a.y * b.y | * | c.x, c.y = a.x * b.x, a.y * b.y | / | c.x, c.y = a.x * b.y, a.y * b.x
Representation	Implementation

Example: Building Desctruction Problem

Problem Definition: A building is about to be blown up. However, there are many expensive devices around which can be damaged by the explosion. We have list their GPS datum, which can be used if they are in the safe distance. If not, return the index of the first unsafe device.

Example: Building Desctruction Problem

Definition Table of Point Data Type

Point	A = (p: Point, x: R) x := p.getX()	mathematical
	A = (p: Point, y: R) x := p.getY()	
	A = (p, q : Point, y: R) d := Distance(p, q)	
x, y: R	x := p.x	algorithmic
	y := p.y	
	d := sqrt((p.x - q.x)^2 + (p.y - q.y)^2)	

Example: Building Desctruction Problem

UML Class Diagram of Point

UML Class Diagram	
Point	
- X: Point - Y: Point	
Point(a:double, b:double) + getX(): double {query (modifies nothing)} + getY(): double {query} + Distance(q: Point): double {query}	+ -> public - -> private # -> protected

Example: Building Desctruction Problem

Definition Table of Circle Data Type

Circle	A = (p: Point, c: Circle, l:L) l := c. Contains(p)
cp: Point r: R r > 0	l := Distance(c.cp, p) < c.r

Example: Building Desctruction Problem

UML Class Diagram of Circle

UML Class Diagram	
Point	
- X: Point - Y: Point	
Point(a:double, b:double) + getX(): double {query (modifies nothing)} + getY(): double {query} + Distance(q: Point): double {query}	+ -> public - -> private # -> protected

Example: Building Desctruction Problem

UML Class Diagram of Circle

Circle
- cp: Point -r: double {r > 0}
+ Circle(cp: Point, r:double) + Contains(p:Point) bool {query}

Example: Building Desctruction Problem Main function

Linear Search

Specification
A = (x: Point^n , c: Circle, l:L, ind: N)
Pre = (x = x' ∧ c = c')
Post = (Pre ∧ l, ind = SEARCH$_{i = 1}^{n}$ c.Contains(x[i])

Analogy

m -> 1
n -> n
Cond(i) -> c.Contains(x[i])

i,l := 1, false	
	i <= n ∧ ¬l
	l, ind = c.Contains(x[i]), i
	i := i + 1