

# MIS Example

## CS 2ME3/SE 2AA4

Sepehr Bayat

Department of Computing and Software  
McMaster University

January 29

# Outline

- 1 What is a module?
- 2 MIS Template
- 3 Example
  - Point ADT Module
  - TriangleADT Module

# What is a module?

- A file with encapsulated code to implement a specific functionality
- Ex: For designing a website with a login system, we may have a module that deals with logging out
- A module comes with an "interface"
- An interface includes things like functions and arguments of the function

# What is a MIS?

- Module Interface Specification
- Specifies externally observable behaviour of a module
- Not in language of implementation, but uses mathematical and application language
- Internal implementations are not included in a MIS

# MIS Template Structure

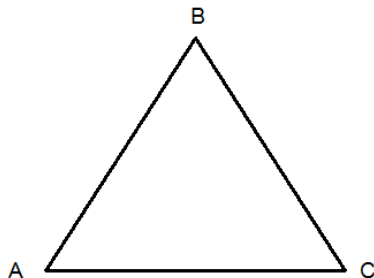
- Uses
  - Imported constants, data types and access programs
- Syntax
  - Exported constants and types
  - Exported functions (access routine interface syntax)
- Semantics
  - State variables
  - State invariants
  - Assumptions
  - Access routine semantics
  - Local functions
  - Local types
  - Local constants
  - Considerations

# Example

Consider implementation of point on two dimensional plane

- Position P is represented by pair of real numbers  $(x,y)$
- A triangle is represented by three points
- Considering we have three points in a 2D surface and we want to know the possibility of having a triangle with the three points and then calculate the perimeter and the area of the triangle.

Consider the following triangle:



We are using the following inequality which is called inequality equation to know the possibility of having triangle with three points A, B and C:



$$AB + AC < BC, AC + BC < AB, AB + BC < AC$$

We are calculating the perimeter of the triangle using the following formula:



$$P = (AB + AC + BC)$$

We are using the Heron formula to calculate the area of the triangle:



$$\sqrt{P/2(P/2 - AB)(P/2 - AC)(P/2 - BC)}$$



# Point ADT Module

## Template Module

pointADT

## Uses

N/A

## Syntax

Exported Types:

pointT = ?

## Exported Access Programs

Routine name	In	Out	Exceptions
init	real, real	pointT	
xcoord		real	
ycoord		real	
dist	pointT	real	

## Semantics

State Variables:

xc: real

yc: real

**State Invariant** None

**Assumptions** init() is called for each abstract object before any other access routine is called for that object

# Point ADT Module

## Access Routine Semantics

`init(x, y):`

- transition:  $xc, yc := x, y$
- output:  $out := self$
- exception: none

`xcoord():`

- output:  $out := xc$
- exception: none

`ycoord():`

- output:  $out := yc$
- exception: none

`dist(p):`

- output:  
$$out := \sqrt{(self.xc - p.xcoord())^2 + (self.yc - p.ycoord())^2}$$
- exception: none

# MIS Interface

From the MIS we can deduce the interface of the code will look like:

```
# Interface
class pointT:

    # Constructor
    def __init__(self, x, y):

    # Selectors
    def xcoord(self):

    def ycoord(self):

    def dist(self, p):
```

# MIS Implementation

See `PointADT.py` for implementation.

# TriangleADT

## Template Module

TriangleADT

## Uses

PointADT

## Syntax

Exported Types:

TriangleADT = ?

## Exported Access Programs

Routine name	In	Out	Exceptions
init	pointT,pointT,pointT	TriangleADT	
sides			
inequality_theorem		boolean	LINE
perimeter_of_triangle		real	LINE
area_of_triangle		real	LINE



## Semantics

State Variables:

$p1$ : pointT

$p2$ : pointT

$p3$ : pointT

$AB$ : real

$AC$  : real

$BC$  : real

**State Invariant** None

**Assumptions** `init()` is called for each abstract object before any other access routine is called for that object

## Access Routine Semantics

`init(p1, p2, p3):`

- transition:  $a, b, c := p1, p2, p3$
- output:  $out := self$
- exception: none

`sides():`

- transition:  $AB, AC, BC := pointT.dist(self.a, self.b), pointT.dist(self.a, self.c), pointT.dist(self.b, self.c)$
- output:  $out := self$
- exception: none

inequality theorem():

■ output:

$$\begin{aligned} out := & (\text{self.AB} + \text{self.AC} > \text{self.BC} \quad \text{and} \quad \text{self.AB} + \text{self.BC} \\ & > \text{self.AC} \quad \text{and} \quad \text{self.AC} + \text{self.BC} > \text{self.AB}) \end{aligned}$$

■ exception:

$$\begin{aligned} ex := & ((\text{self.a.xcoord()} == \text{self.b.xcoord()} == \text{self.c.xcoord()} \text{ or} \\ & \text{self.a.ycoord()} == \text{self.b.ycoord()} == \text{self.c.ycoord()})) \Rightarrow \text{LINE}) \end{aligned}$$

perimeter of triangle():

- output:

$$out := self.AB + self.AC + self.BC$$

- exception:

ex := ((self.a.xcoord()==self.b.xcoord() ==self.c.xcoord() or  
self.a.ycoord()==self.b.ycoord() ==self.c.ycoord()))  $\Rightarrow$  LINE)

area of triangle():

- output :

$$out := \sqrt{P/2(P/2 - self.AB)(P/2 - self.AC)(P/2 - self.BC)}$$

- exception:

ex := ((self.a.xcoord()==self.b.xcoord() ==self.c.xcoord() or  
self.a.ycoord()==self.b.ycoord() ==self.c.ycoord()))  $\Rightarrow$  LINE)

## Local Constants

P := self.AB+self.AC+self.BC

# MIS Interface

From the MIS we can deduce the interface of the code will look like:

```
# Interface
class TriangleADT:

    #Constructor
    def __init__(self, p1, p2, p3):

    #Selectors
    def sides(self):

    def inequality_theorem(self):

    def perimeter_of_triangle(self):

    def area_of_triangle(self):
```

# MIS Implementation

See TriangleADT for implementation.

# Implementation files

- Implementation files PointADT.py and TriangleADT.py can be found in the repo under Tutorial/T4/src