SE 2AA4, CS 2ME3 (Introduction to Software Development)

Winter 2018

28 Parnas Tables

Dr. Spencer Smith

Faculty of Engineering, McMaster University

March 20, 2018

28 Parnas Tables

- Today's slide are partially based on slides by Dr. Wassyng
- Administrative details
- Design patterns
- Motivating example: midterm question
- History of tables
- Example tables
- Semantics for tables
- Classification of tables
- Tables in practise
- Advantages of tables
- pointInRegion(p)

Administrative Details

- Multiple potential Academic Dishonesty cases are currently being investigated
- A3
  - Part 2 - Code: due 11:59 pm Mar 26
- A4
  - Due April 9 at 11:59 pm

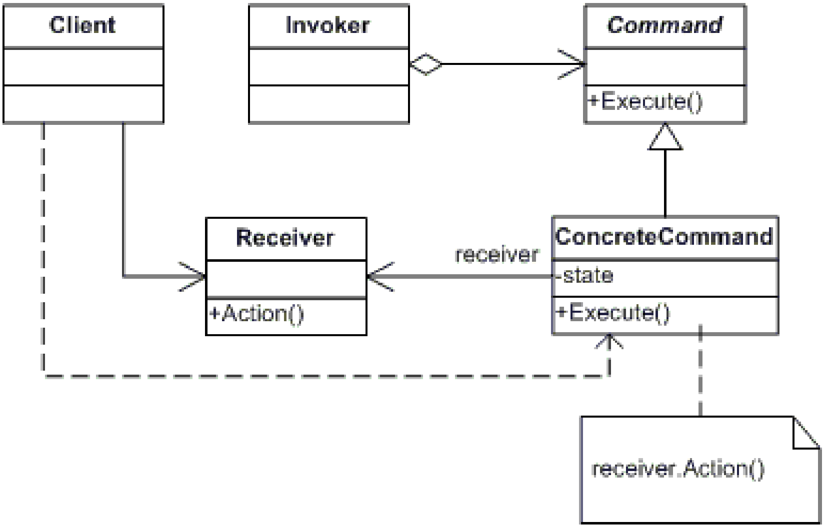
Command Processor Pattern

- Context: User interfaces which must be flexible or provide functionality that goes beyond the direct handling of user functions. Examples are undo facilities or logging functions
- Problem: We want a well-structured solution for mapping an interface to the internal functionality of a system. All 'extras' which have to do with the way user commands are input, additional commands such as undo and redo, and any non-application-specific processing of user commands, such as logging, should be kept separate from the interface to the internal functionality.

Command Processor Pattern Continued

- Solution: A separate component, the command processor, takes care of all commands. The command processor component schedules the execution of commands, stores them for later undo, logs them for later analysis, and so on. The actual execution of the command is delegated to a supplier component within the application.

Command in UML

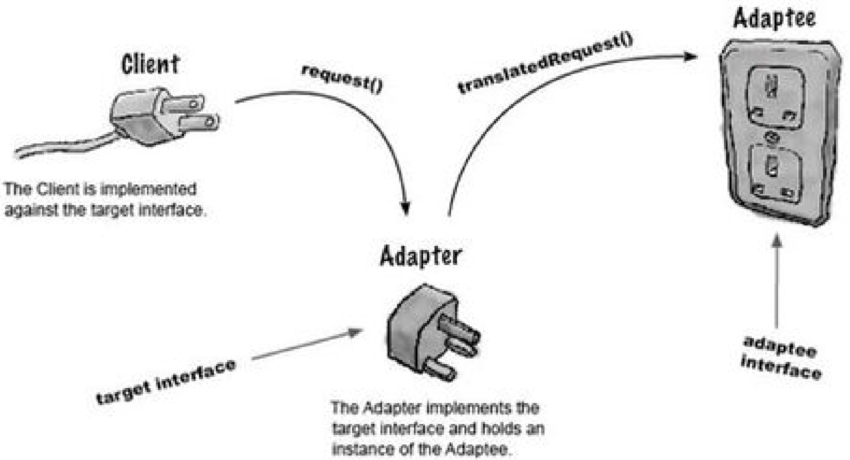


http://www.dofactory.com/net/command-design-pattern

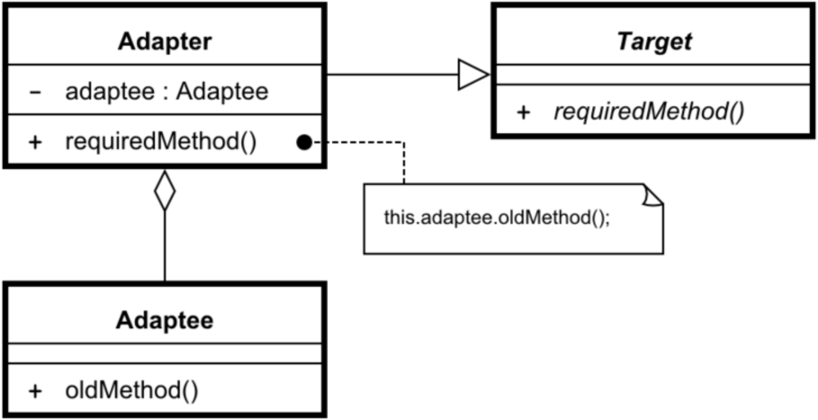
Adapter Design Pattern

When have we used the adapter (or wrapper) design pattern?

Adapter Design Pattern

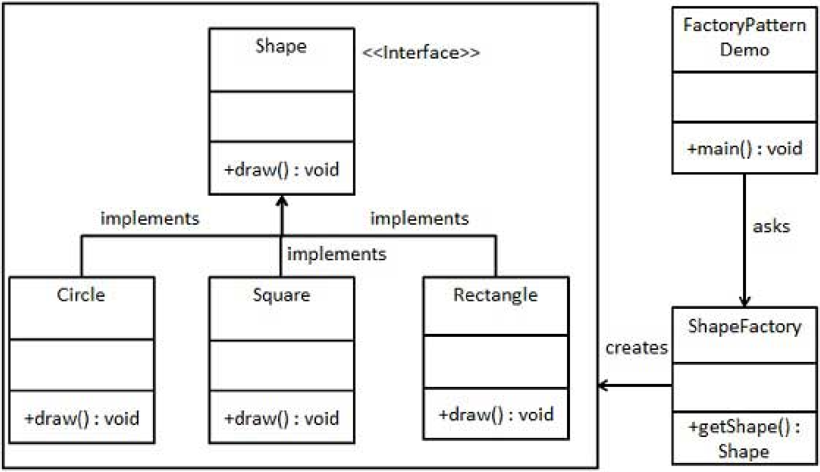


Adapter UML Diagram



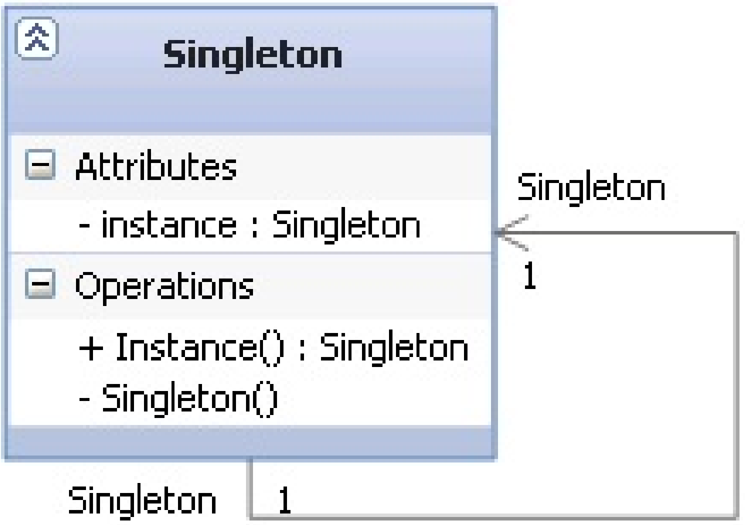
Link

Factory Pattern



Code

Singleton Pattern



Tables Motivating Example: ptOn(c, s)

$$
\begin{aligned}
(x(c) = x(s) \Rightarrow\ & (y(s) \leq y(f) \Rightarrow y(s) \leq y(c) \leq y(f) | \\
& \ y(s) > y(f) \Rightarrow y(f) \leq y(c) \leq y(s)) \\
| y(c) = y(s) \Rightarrow\ & (x(s) \leq x(f) \Rightarrow x(s) \leq x(c) \leq x(f) | \\
& \ x(s) > x(f) \Rightarrow x(f) \leq x(c) \leq x(s)) \\
| N(c, s) \Rightarrow\ & \text{False})
\end{aligned}
$$

$$N(c, s) \equiv x(c) \neq x(s) \wedge y(c) \neq y(s)$$

In Tabular Form

		out
$x(c) = x(s)$	$y(s) \leq y(f)$	$y(s) \leq y(c) \leq y(f)$
	$y(s) > y(f)$	$y(f) \leq y(c) \leq y(s)$
$y(c) = y(s)$	$x(s) \leq x(f)$	$x(s) \leq x(c) \leq x(f)$
	$x(s) > x(f)$	$x(f) \leq x(c) \leq x(s)$
$x(c) \neq x(s) \wedge y(c) \neq y(s)$		False

A Brief History of Tables

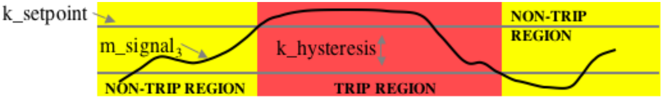
- Similar work, such as decision tables, have been around for a while (1950s?)
- The intuitive use of tables proliferated on the A-7E Aircraft US Naval Research Lab (NRL) project (Parnas)
- The US NRL continues to work on the SCR (Software Cost Reduction) method
- Ontario Hydro - Darlington Shutdown Systems
- Work began on the semantics of tables - Parnas, Janicki, Zucker, Abraham
- Ontario Power Generation (OPG) methods for Safety Critical Software
- More semantics - Janicki, Khedri, Kahl, Wassyng
- Dave Parnas has championed the use of tabular expressions (tables) in documenting software requirements and designs

Example Table from A-7E Project

SCR/A7-E

Inmode(*High*) ∨ Inmode(*Permitted*)	$TRUE$	$FALSE$
Inmode(*TooLow*)	!Overridden!	¬!Overridden!
%%Safety_Injection%% =	OFF	ON

Example Table from OPG Project



OPG

f_sensortrip$_i$, i=1,..,4

{For each i = 1,..,4}

Condition	Result **f_sensortrip$_i$**
k_setpoint ≤ m_signal$_i$ *{i^{th} signal is now in the trip region}*	e_Trip
k_setpoint - k_hysteresis < m_signal$_i$ < k_setpoint *{i^{th} signal is now in the deadband region}*	No Change
m_signal$_i$ ≤ k_setpoint - k_hysteresis *{i^{th} signal is now in the non-trip region}*	e_NotTrip

What is this table specifying?

Example for Input Checking

Composition rule	$\cup_{i=1}^{4} H_2[i] \cap (\cap_{j=1}^{2} H_1[j] \; ; \; G[i,j])$

H_1

$S'_{GET} \cup =$	$ErrorMsg' + =$

H_2	G	
$x_1 < 0$	$\emptyset$	$InvalidInput\_x_1$
$0 \leq x_1 < min_d$	$\emptyset$	$x_1\_TooSmall$
$x_1 > max_d$	$\emptyset$	$x_1\_TooLarge$
$min_d \leq x_1 \leq max_d$	$\{@x_1\}$	$NULL$

$\wedge ChangeOnly(S_{GET}, ErrorMsg)$

H_2 $\qquad$ G

Solving Real Roots of $ax^2 + bx + c = 0$

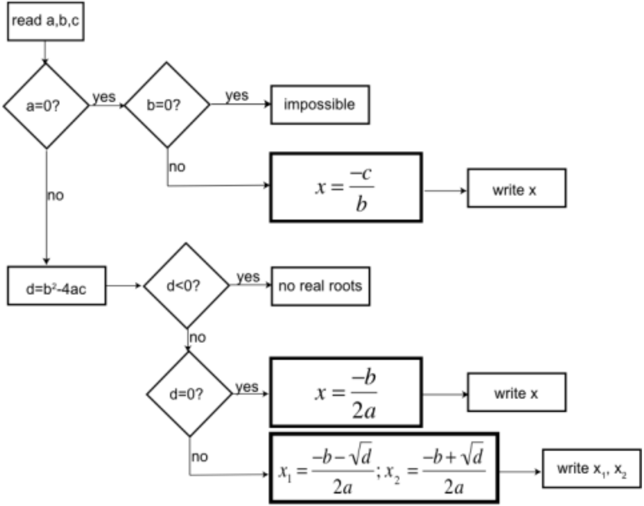


Table for Solving the Quadratic Equation

			Solution caption	x_1	x_2
$a = 0$	$b = 0$		Ill defined	—	—
	$b \neq 0$		One root	$-\frac{c}{b}$	—
$a \neq 0$	$b^2 - 4ac < 0$		No real roots	—	—
	$b^2 - 4ac = 0$		One root	$-\frac{b}{2a}$	—
	$b^2 - 4ac > 0$	$b \geq 0$	Two roots	$\frac{-b-\sqrt{b^2-4ac}}{2a}$	$\frac{c}{ax_1}$
		$b < 0$	Two roots	$\frac{-b+\sqrt{b^2-4ac}}{2a}$	$\frac{c}{ax_1}$

What are the advantages of the tabular specification?

Table for Solving the Quadratic Equation

			Solution caption	x_1	x_2
$a = 0$	$b = 0$		Ill defined	—	—
	$b \neq 0$		One root	$-\frac{c}{b}$	—
$a \neq 0$	$b^2 - 4ac < 0$		No real roots	—	—
	$b^2 - 4ac = 0$		One root	$-\frac{b}{2a}$	—
	$b^2 - 4ac > 0$	$b \geq 0$	Two roots	$\frac{-b-\sqrt{b^2-4ac}}{2a}$	$\frac{c}{ax_1}$
		$b < 0$	Two roots	$\frac{-b+\sqrt{b^2-4ac}}{2a}$	$\frac{c}{ax_1}$

What are the advantages of the tabular specification?

- Understandable
- Unambiguous
- Check for completeness and disjointness
- Test cases

Why We Need Precise Semantics

- To promote an unambiguous understanding for both writers and readers
- To understand the meaning of tables that look similar, but have different semantics
- To be able to link tables of different types
- To know what notation we can use in the tables
- To be able to build software tools that create, edit, transform and print tables

Early Semantics

	f_name
Condition 1	Result 1
Condition 2	Result 2
...	...
Condition n	Result n

If Condition 1 then f_name = Result 1
Elseif Condition 2 then f_name = Result 2
Elseif ...
Elseif Condition n then f_name = Result n
or f_name = (Condition 1 $\Rightarrow$ Result 1 | ...)

Early Semantics

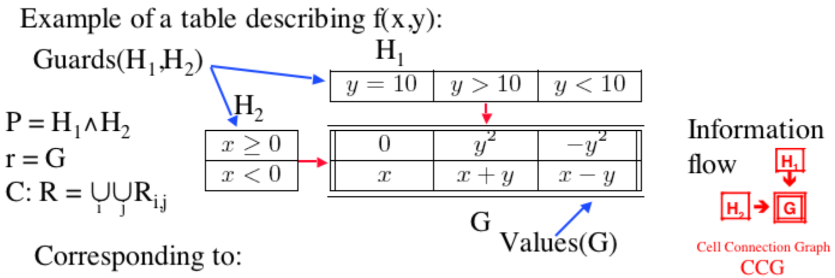
	f_name
Condition 1	Result 1
Condition 2	Result 2
...	...
Condition n	Result n

If Condition 1 then f_name = Result 1
Elseif Condition 2 then f_name = Result 2
Elseif ...
Elseif Condition n then f_name = Result n
or f_name = (Condition 1 $\Rightarrow$ Result 1 | ...)

Disjointedness $\equiv \forall(i, j : \mathbb{N} | 1 \leq i \leq n \wedge 1 \leq j \leq n \wedge i \neq j :$ Condition $i \wedge$ Condition $j \Leftrightarrow$ false$)$

Completeness $\equiv \vee(i : \mathbb{N} | 1 \leq i \leq n :$ Condition $i)$

Semantics



$$f(x,y) = \begin{cases} 0 & \text{if } x \geq 0 \wedge y = 10 \\ x & \text{if } x < 0 \wedge y = 10 \\ y^2 & \text{if } x \geq 0 \wedge y > 10 \\ -y^2 & \text{if } x \geq 0 \wedge y < 10 \\ x + y & \text{if } x < 0 \wedge y > 10 \\ x - y & \text{if } x < 0 \wedge y < 10 \end{cases}$$

Semantics Continued

- Disjointedness and Completness are not part of the semantics of tables
- We impose these conditions to make tables more useful in practise

Classification of Tables

Tabular expressions can be classified according to the orientation of the tables

Vertical condition tables

	c1	c2
res	v1	v2

Horizontal condition tables

	res
c1	v1
c2	v2

Types of Tables

Normal

	$y = 10$	$y > 10$	$y < 10$
$x \geq 0$	0	y^2	$-y^2$
$x < 0$	x	$x + y$	$x - y$

value cells

Inverted

	$x + y$	$x - y$	$y - x$
$y \geq 0$	$x < 0$	$0 \leq x < y$	$x \geq y$
$y < 0$	$x < y$	$y \leq x < 0$	$x \geq 0$

Decision

value cells

	gs	gb	gb	pb	g
Tem $\in \{h, c\}$	$*$	$*$	h	$*$	c
Wt $\in \{s, cl, r\}$	$s \vee cl$	s	cl	r	cl
Wn $\in \{T, F\}$	T	F	F	$*$	F

$* = don't\ care,\ T = true,\ F = false$
$h = hot,\ c = cool,\ s = sunny,\ cl = cloudy,$
$r = rain$
$Tem = Temperature,\ Wt = Weather,$
$Wn = Windy$
$gs = go\ sailing,\ gb = go\ to\ the\ beach$
$pb = play\ bridge,\ g = garden$

Vector

	$x_2 \leq 0$	$x_2 > 0$
$y_1 =$	$x_1 + x_2$	$x_1 - x_2$
y_2	$y_2 x_1 - x_2 = y_2^2$	$x_1 + x_2 y_2 = \|y_2\|$
y_3	$y_3 + x_1 x_2 = \|y_3\|^3$	$y_3 = x_1$

value cells

Generalized Decision

value cells

	$x_1 + x_2$	$x_1 - x_2$	$x_1 x_2$
$x_1 x_2$	$\# < 20$	$\# \geq 20$	*true*
$x_1 \div x_2$	$\# > 30$	$\# < 30$	$\# = 30$

World View of Tables

- Do tables take a dynamic or a static world view?
- Can you directly write an algorithm from a table?

Tables in Practise

- According to Dr. Wassyng projects typically define a small set of types of tables to be used in that project
- Tables at Ontario Power Generation, Darlington Nuclear Generating Station - Shutdown System One (SDS1)
  - Horizontal condition tables for requirements - read from left to right, fit on the page well
  - Vertical condition tables for the software design - better suited to multiple outputs
  - Sometimes also state transition tables
- Use table structure to visually aid readers so that they can discern nested conditions (see the quadratic equation example)
- Tables enable production of formal requirements that are readable by domain experts
- Use natural language expressions to enhance readability

Advantages of Tables

- Tabular expressions describe relations through pre and post conditions - ideal for describing behaviour without sequences of operations
- They make it easy to ensure input domain coverage
- They are easy to read and understand (you need just a little practise to write them)
- Coding from tables results in extremely well structured code
- They facilitate identification of test cases
- Extremely good for real-time/embedded systems

A Table for pointInRegion(p)

- Consider all of the cases for a rectangle
- Draw a picture
- Short form notation
  - $px = p.\text{xcoord}()$
  - $py = p.\text{ycoord}()$
  - $llx = lower\_left.\text{xcoord}()$
  - $lly = lower\_left.\text{ycoord}()$
  - $llxw = lower\_left.\text{xcoord}() + width$
  - $llyh = lower\_left.\text{ycoord}() + height$
  - $\text{T} = \text{Constants.TOLERANCE}$
  - $p1.\text{dist}(p2)$ is the distance between p1 and p2

		out

		out
px $<$ *llx*		
llx $\leq$ *px* $\leq$ *llxw*		
px $>$ *llxw*		

		out
$px < llx$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	
$llx \leq px \leq llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	
$px > llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	

		out
$px < llx$	$py < lly$	$p.\mathrm{dist}(\mathrm{PointT}(llx, lly)) \leq \mathrm{T}$
	$lly \leq py \leq llyh$	
	$py > llyh$	
$llx \leq px \leq llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	
$px > llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	

		out
$px < llx$	$py < lly$	$p.\mathrm{dist}(\mathrm{PointT}(llx, lly)) \leq \mathrm{T}$
	$lly \leq py \leq llyh$	$(llx - px) \leq \mathrm{T}$
	$py > llyh$	
$llx \leq px \leq llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	
$px > llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	

		out
$px < llx$	$py < lly$	$p.\text{dist}(\text{PointT}(llx, lly)) \leq \text{T}$
	$lly \leq py \leq llyh$	$(llx - px) \leq \text{T}$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llx, llyh)) \leq \text{T}$
$llx \leq px \leq llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	
$px > llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	

		out
$px < llx$	$py < lly$	$p.\text{dist}(\text{PointT}(llx, lly)) \leq \text{T}$
	$lly \leq py \leq llyh$	$(llx - px) \leq \text{T}$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llx, llyh)) \leq \text{T}$
$llx \leq px \leq llxw$	$py < lly$	$(lly - py) \leq \text{T}$
	$lly \leq py \leq llyh$	
	$py > llyh$	
$px > llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	

		out
$px < llx$	$py < lly$	$p.\text{dist}(\text{PointT}(llx, lly)) \leq \text{T}$
	$lly \leq py \leq llyh$	$(llx - px) \leq \text{T}$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llx, llyh)) \leq \text{T}$
$llx \leq px \leq llxw$	$py < lly$	$(lly - py) \leq \text{T}$
	$lly \leq py \leq llyh$	True
	$py > llyh$	
$px > llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	

		out
$px < llx$	$py < lly$	$p.\mathrm{dist}(\mathrm{PointT}(llx, lly)) \leq \mathrm{T}$
	$lly \leq py \leq llyh$	$(llx - px) \leq \mathrm{T}$
	$py > llyh$	$p.\mathrm{dist}(\mathrm{PointT}(llx, llyh)) \leq \mathrm{T}$
$llx \leq px \leq llxw$	$py < lly$	$(lly - py) \leq \mathrm{T}$
	$lly \leq py \leq llyh$	True
	$py > llyh$	$(py - llyh) \leq \mathrm{T}$
$px > llxw$	$py < lly$	
	$lly \leq py \leq llyh$	
	$py > llyh$	

		out
$px < llx$	$py < lly$	$p.\text{dist}(\text{PointT}(llx, lly)) \leq \text{T}$
	$lly \leq py \leq llyh$	$(llx - px) \leq \text{T}$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llx, llyh)) \leq \text{T}$
$llx \leq px \leq llxw$	$py < lly$	$(lly - py) \leq \text{T}$
	$lly \leq py \leq llyh$	True
	$py > llyh$	$(py - llyh) \leq \text{T}$
$px > llxw$	$py < lly$	$p.\text{dist}(\text{PointT}(llxw, lly)) \leq \text{T}$
	$lly \leq py \leq llyh$	$(px - llxw) \leq \text{T}$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llxw, llyh)) \leq \text{T}$

Seven Cases

		out
$px < llx$	$py < lly$	$p.\text{dist}(\text{PointT}(llx, lly)) \leq \text{T}$
	$lly \leq py \leq llyh$	$(llx - px) \leq \text{T}$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llx, llyh)) \leq \text{T}$
$llx \leq px \leq llxw$		
$px > llxw$	$py < lly$	$p.\text{dist}(\text{PointT}(llxw, lly)) \leq \text{T}$
	$lly \leq py \leq llyh$	$(px - llxw) \leq \text{T}$
	$py > llyh$	$p.\text{dist}(\text{PointT}(llxw, llyh)) \leq \text{T}$

Seven Cases

		out
$px < llx$	$py < lly$	$p.\mathrm{dist}(\mathrm{PointT}(llx, lly)) \leq \mathrm{T}$
	$lly \leq py \leq llyh$	$(llx - px) \leq \mathrm{T}$
	$py > llyh$	$p.\mathrm{dist}(\mathrm{PointT}(llx, llyh)) \leq \mathrm{T}$
$llx \leq px \leq llxw$		$(lly - \mathrm{T}) \leq py \leq (llyh + \mathrm{T})$
$px > llxw$	$py < lly$	$p.\mathrm{dist}(\mathrm{PointT}(llxw, lly)) \leq \mathrm{T}$
	$lly \leq py \leq llyh$	$(px - llxw) \leq \mathrm{T}$
	$py > llyh$	$p.\mathrm{dist}(\mathrm{PointT}(llxw, llyh)) \leq \mathrm{T}$

Six Cases

		out
$px < llx$	$py < lly$	$p.\mathrm{dist}(\mathrm{PointT}(llx, lly)) \leq \mathrm{T}$
	$py > llyh$	$p.\mathrm{dist}(\mathrm{PointT}(llx, llyh)) \leq \mathrm{T}$
$llx \leq px \leq llxw$		$(lly - \mathrm{T}) \leq py \leq (llyh + \mathrm{T})$
$px > llxw$	$py < lly$	$p.\mathrm{dist}(\mathrm{PointT}(llxw, lly)) \leq \mathrm{T}$
	$py > llyh$	$p.\mathrm{dist}(\mathrm{PointT}(llxw, llyh)) \leq \mathrm{T}$
$lly \leq py \leq llyh$		

Six Cases

		out
$px < llx$	$py < lly$	$p.\mathrm{dist}(\mathrm{PointT}(llx, lly)) \leq \mathrm{T}$
	$py > llyh$	$p.\mathrm{dist}(\mathrm{PointT}(llx, llyh)) \leq \mathrm{T}$
$llx \leq px \leq llxw$		$(lly - \mathrm{T}) \leq py \leq (llyh + \mathrm{T})$
$px > llxw$	$py < lly$	$p.\mathrm{dist}(\mathrm{PointT}(llxw, lly)) \leq \mathrm{T}$
	$py > llyh$	$p.\mathrm{dist}(\mathrm{PointT}(llxw, llyh)) \leq \mathrm{T}$
$lly \leq py \leq llyh$		$(llx - \mathrm{T}) \leq px \leq (llxw + \mathrm{T})$

Three Cases

	out
$llx \leq px \leq llxw$	$(lly - \mathrm{T}) \leq py \leq (llyh + \mathrm{T})$
$lly \leq py \leq llyh$	$(llx - \mathrm{T}) \leq px \leq (llxw + \mathrm{T})$
$\neg(llx \leq px \leq llxw) \wedge \neg(lly \leq py \leq llyh)$	

Three Cases

	out
$llx \leq px \leq llxw$	$(lly - \mathrm{T}) \leq py \leq (llyh + \mathrm{T})$
$lly \leq py \leq llyh$	$(llx - \mathrm{T}) \leq px \leq (llxw + \mathrm{T})$
$\neg(llx \leq px \leq llxw) \wedge \neg(lly \leq py \leq llyh)$	$\min[p.\mathrm{dist}(\mathrm{PointT}(llx, lly)), p.\mathrm{dist}(\mathrm{PointT}(llxw, lly)), p.\mathrm{dist}(\mathrm{PointT}(llx, llyh)), p.\mathrm{dist}(\mathrm{PointT}(llxw, llyh))] \leq \mathrm{T}$

Nine Cases, but 2D

- How would you write all 9 cases, but with a tabular form that closely matches the original 2D problem description?