EiffelStudio – the Eiffel IDE

Christian Estler
ETH Zurich
christian.estler@inf.ethz.ch

Distributed and Outsourced Software Engineering - ETH course, Fall 2012

EiffelStudio – an Overview

- EiffelStudio (ES) is an Integrated Development Environment to write Eiffel programs
  - Developed by Eiffel Software
  - First version ca. 1990
  - Current version is 7.1
  - Available on SourceForge

- We use ES in the course

- Many things are similar to IDEs like Eclipse or VisualStudio; some things are different or tricky→ that's what we'll talk about here

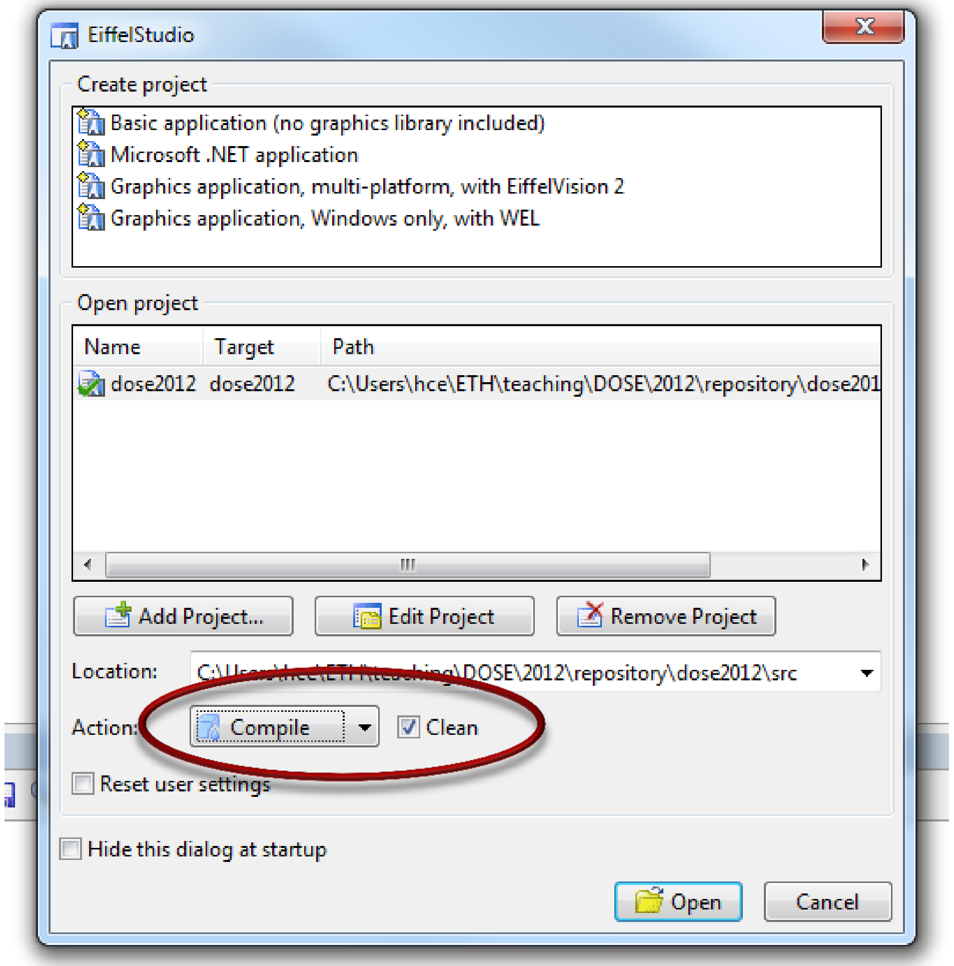
EiffelStudio – an Overview

We will talk about

- *Clean compile* (what is it, why is it needed?)
- Code Browsing
- Code Views
- EiffelStudio’s navigation using *Pick & Drop*
- Debugging in EiffelStudio
- Auto-completion and Code-Templates
- Refactoring tools

Clean Compile

- ***Clean compile*** means compiling the project from scratch
- Necessary e.g. after adding or removing libraries
- Deletes previously generated intermediate compile information (the EIFGENs folder)
- Rule of thumb: if your program shows strange behavior, do a clean compile


EiffelStudio

Create project

- Basic application (no graphics library included)
- Microsoft .NET application
- Graphics application, multi-platform, with EiffelVision 2
- Graphics application, Windows only, with WEL

Open project

Name	Target	Path
dose2012	dose2012	C:\Users\hce\ETH\teaching\DOSE\2012\repository\dose201

Location: C:\Users\hce\ETH\teaching\DOSE\2012\repository\dose2012\src

Action: Compile [✓ Clean]

Reset user settings

Hide this dialog at startup

If you remember only one thing from this talk, it should be **clean compile**

more: http://docs.eiffel.com/book/eiffelstudio/clean-compile

Code Browsing

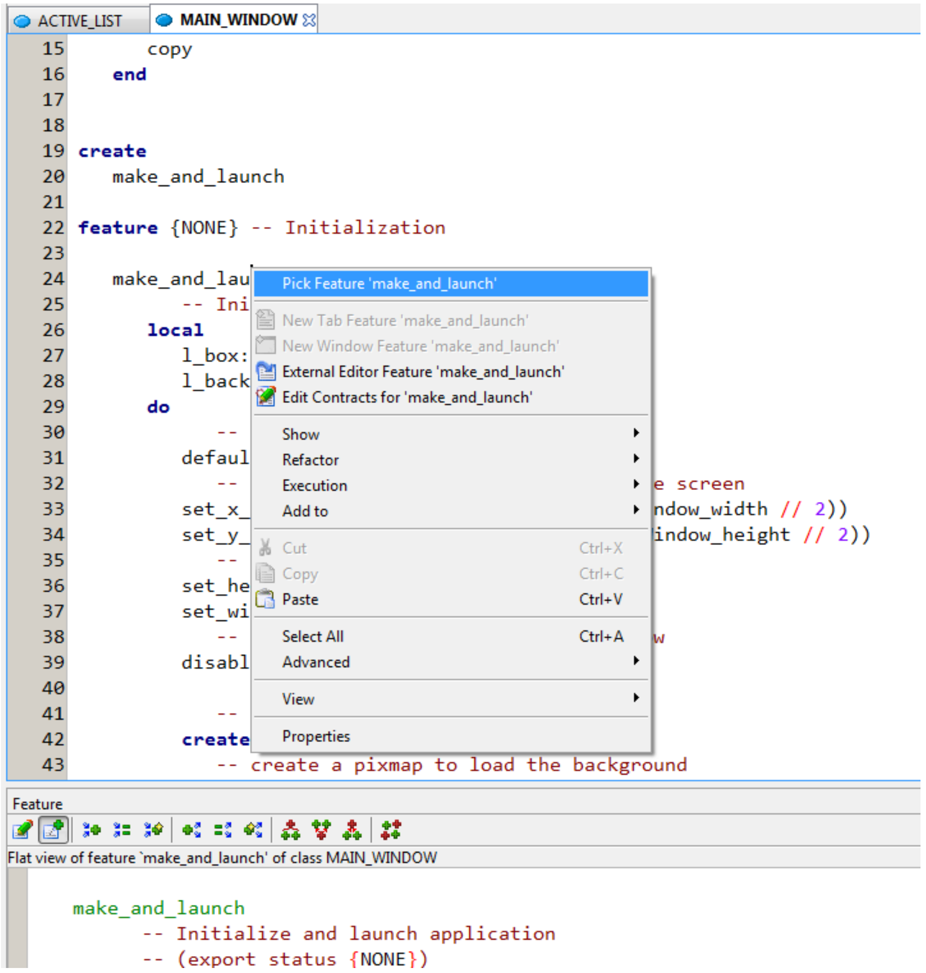
- ES has many features for browsing code; you'll often use the following:

- Group & Feature View

- Class tool to determine
  - Ancestors
  - Descendants
  - Clients

- Feature tool to determine
  - Flat view
  - Callers
  - Callees
  - …

- Navigation through ***Pick & Drop***
  (ES way of doing Drag & Drop)


ACTIVE_LIST
MAIN_WINDOW
Pick Feature 'make_and_launch'
New Tab Feature 'make_and_launch'
New Window Feature 'make_and_launch'
External Editor Feature 'make_and_launch'
Edit Contracts for 'make_and_launch'
Show
Refactor
Execution
Add to
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Select All Ctrl+A
Advanced
View
Properties
Feature

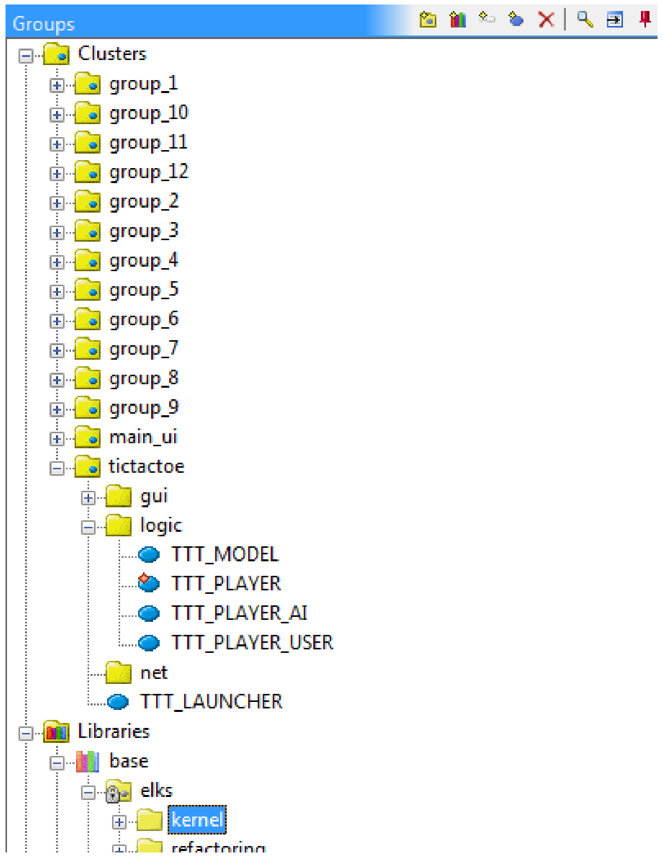
```
        copy
    end


create
    make_and_launch

feature {NONE} -- Initialization

    make_and_lau
        -- Ini
        local
            l_box:
            l_back
        do
            --
            defaul
            --                                  e screen
            set_x_                              ndow_width // 2))
            set_y_                              indow_height // 2))
            --
            set_he
            set_wi
            --                                  w
            disabl

            --
            create
                -- create a pixmap to load the background
```

Flat view of feature `make_and_launch' of class MAIN_WINDOW

```
make_and_launch
        -- Initialize and launch application
        -- (export status {NONE})
```

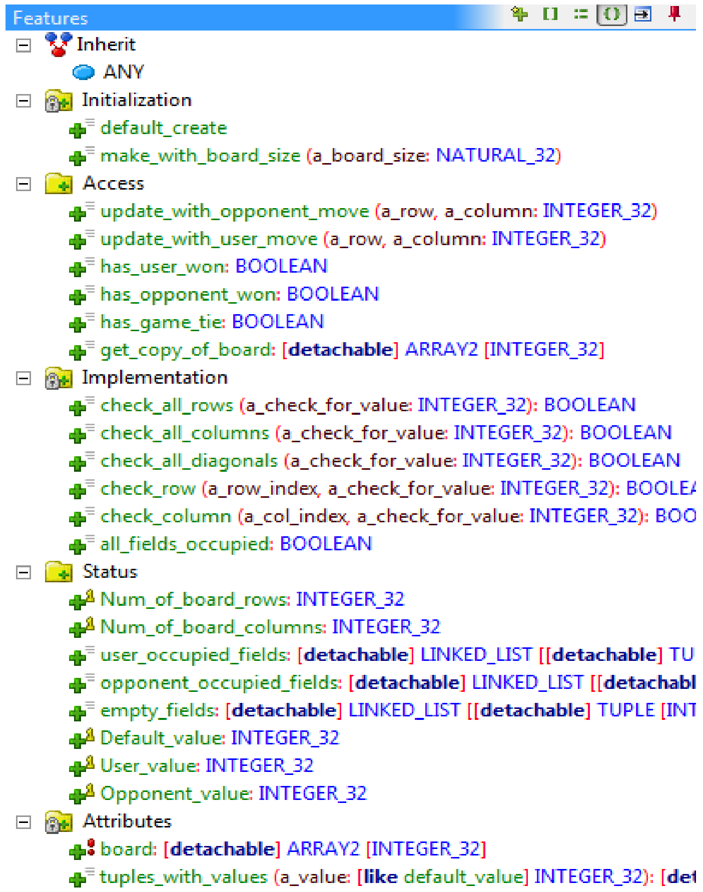
Code Browsing – the Basics

- Group View



```
Groups
Clusters
  group_1
  group_10
  group_11
  group_12
  group_2
  group_3
  group_4
  group_5
  group_6
  group_7
  group_8
  group_9
  main_ui
  tictactoe
    gui
    logic
      TTT_MODEL
      TTT_PLAYER
      TTT_PLAYER_AI
      TTT_PLAYER_USER
    net
    TTT_LAUNCHER
Libraries
  base
    elks
      kernel
      refactoring
```

- Feature View



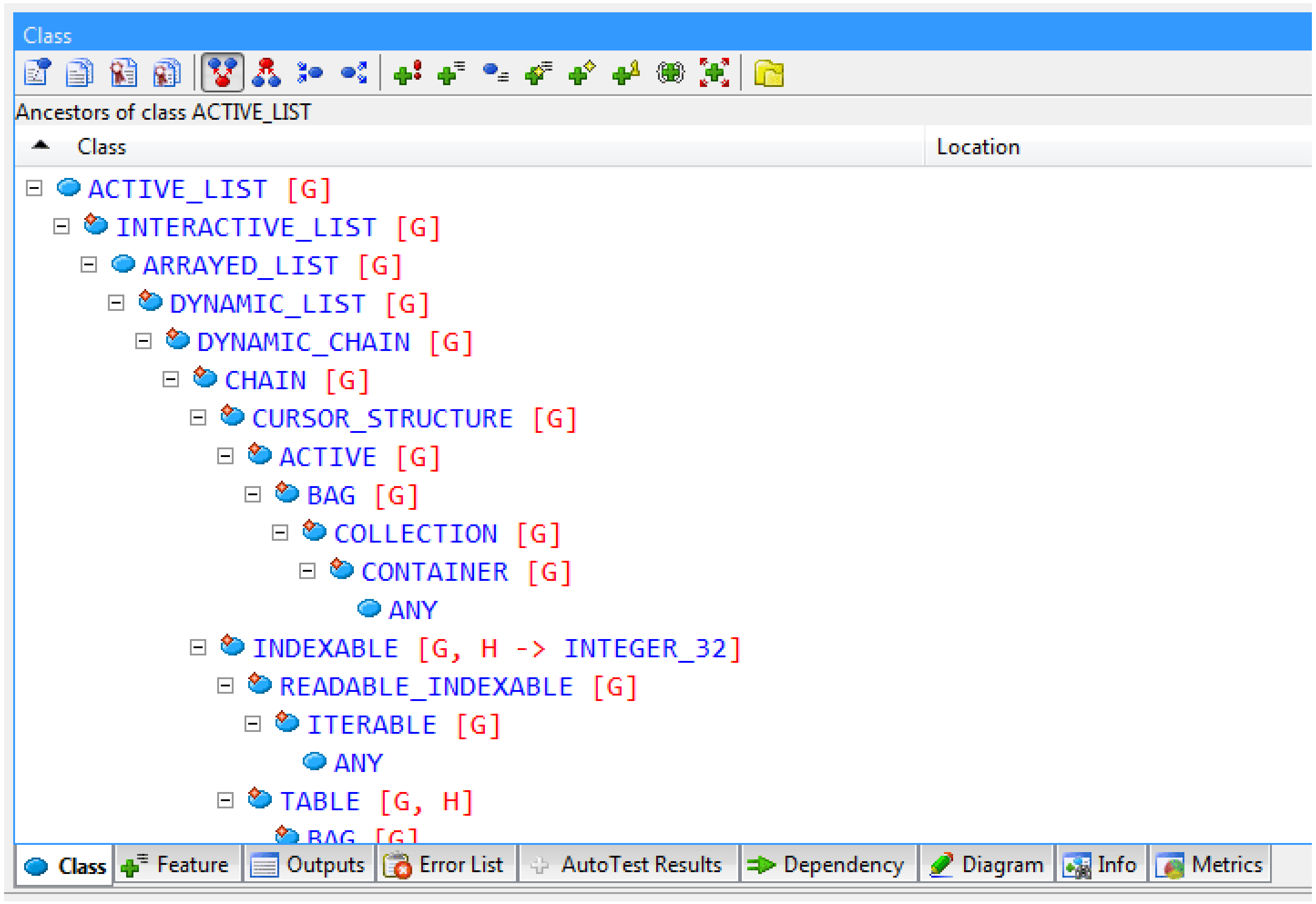
```
Features
Inherit
  ANY
Initialization
  default_create
  make_with_board_size (a_board_size: NATURAL_32)
Access
  update_with_opponent_move (a_row, a_column: INTEGER_32)
  update_with_user_move (a_row, a_column: INTEGER_32)
  has_user_won: BOOLEAN
  has_opponent_won: BOOLEAN
  has_game_tie: BOOLEAN
  get_copy_of_board: [detachable] ARRAY2 [INTEGER_32]
Implementation
  check_all_rows (a_check_for_value: INTEGER_32): BOOLEAN
  check_all_columns (a_check_for_value: INTEGER_32): BOOLEAN
  check_all_diagonals (a_check_for_value: INTEGER_32): BOOLEAN
  check_row (a_row_index, a_check_for_value: INTEGER_32): BOOLEA
  check_column (a_col_index, a_check_for_value: INTEGER_32): BOO
  all_fields_occupied: BOOLEAN
Status
  Num_of_board_rows: INTEGER_32
  Num_of_board_columns: INTEGER_32
  user_occupied_fields: [detachable] LINKED_LIST [[detachable] TU
  opponent_occupied_fields: [detachable] LINKED_LIST [[detachabl
  empty_fields: [detachable] LINKED_LIST [[detachable] TUPLE [INT
  Default_value: INTEGER_32
  User_value: INTEGER_32
  Opponent_value: INTEGER_32
Attributes
  board: [detachable] ARRAY2 [INTEGER_32]
  tuples_with_values (a_value: [like default_value] INTEGER_32): [det
```

Tip 1: arrange Views such that both are visible all the time. You'll use them a lot.

Tip 2: Pick & Drop also works with these Views. Alternative to double-click.

Code Browsing – the Class Tool

- Eiffel makes have use of (multiple) inheritance
- Class tool provides all information on the class level



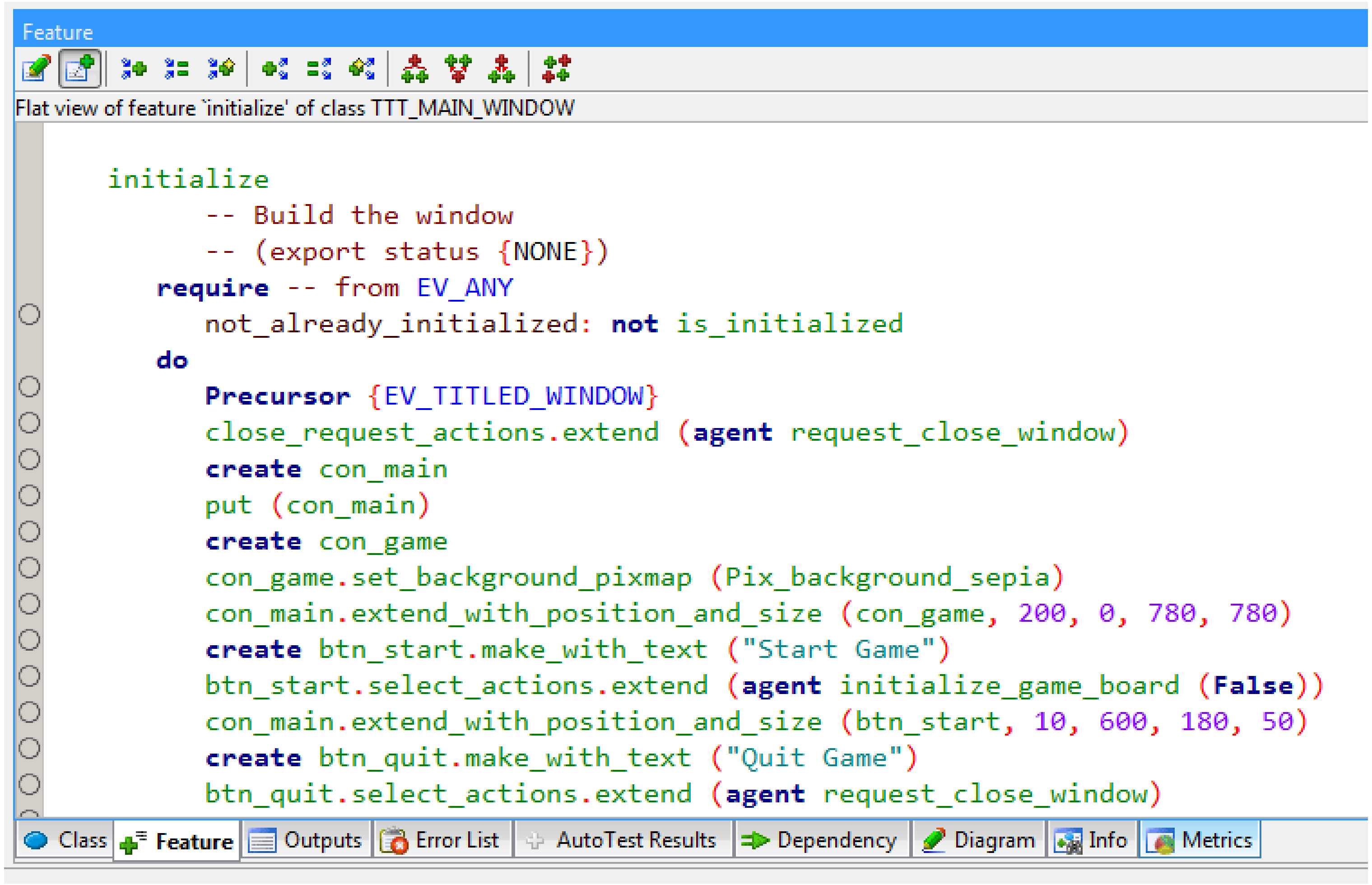
Class

Ancestors of class ACTIVE_LIST

```
Class                                                   Location
ACTIVE_LIST [G]
  INTERACTIVE_LIST [G]
    ARRAYED_LIST [G]
      DYNAMIC_LIST [G]
        DYNAMIC_CHAIN [G]
          CHAIN [G]
            CURSOR_STRUCTURE [G]
              ACTIVE [G]
                BAG [G]
                  COLLECTION [G]
                    CONTAINER [G]
                      ANY
            INDEXABLE [G, H -> INTEGER_32]
              READABLE_INDEXABLE [G]
                ITERABLE [G]
                  ANY
              TABLE [G, H]
                BAG [G]
```

Code Browsing – the Feature Tool

- Feature tool provides all information of a feature



Flat view of feature `initialize' of class TTT_MAIN_WINDOW

```
initialize
		-- Build the window
		-- (export status {NONE})
	require -- from EV_ANY
		not_already_initialized: not is_initialized
	do
		Precursor {EV_TITLED_WINDOW}
		close_request_actions.extend (agent request_close_window)
		create con_main
		put (con_main)
		create con_game
		con_game.set_background_pixmap (Pix_background_sepia)
		con_main.extend_with_position_and_size (con_game, 200, 0, 780, 780)
		create btn_start.make_with_text ("Start Game")
		btn_start.select_actions.extend (agent initialize_game_board (False))
		con_main.extend_with_position_and_size (btn_start, 10, 600, 180, 50)
		create btn_quit.make_with_text ("Quit Game")
		btn_quit.select_actions.extend (agent request_close_window)
```

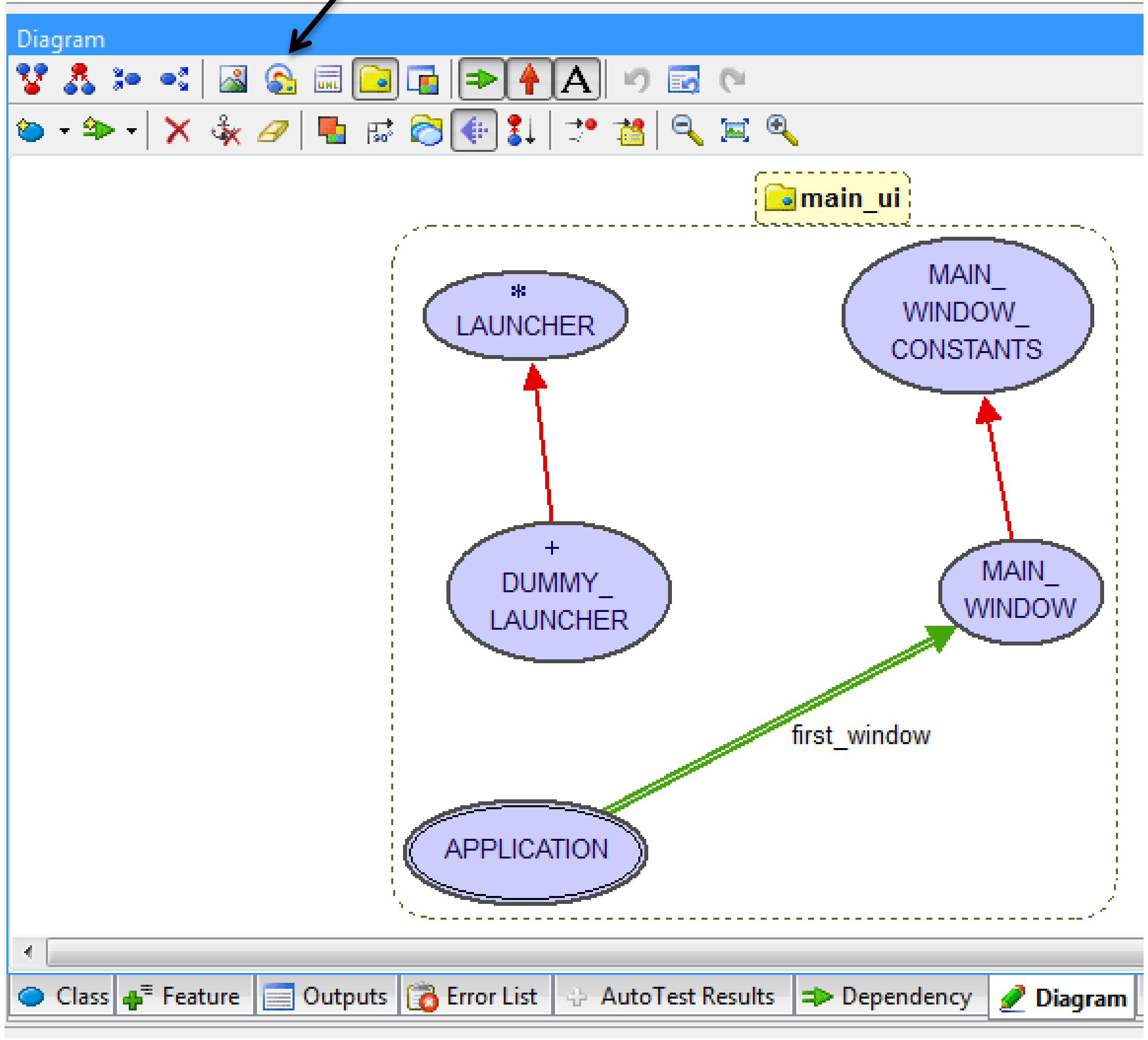
Tip: Pick & Drop also works within the Class and Feature tools (e.g. pick something inside the Feature tool and drop it right there)

Code Browsing – the Diagram Tool

- Diagram tool can be used to a high-level overview of the entire system (more than Class and Feature tool)

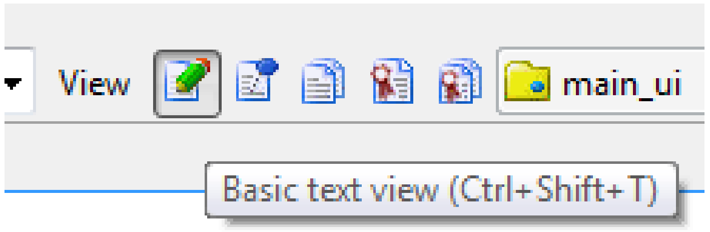
- Pick & Drop a cluster on the diagram target icon

Target Icon



Code Browsing – Different Code Views

- You can inspect code in different views

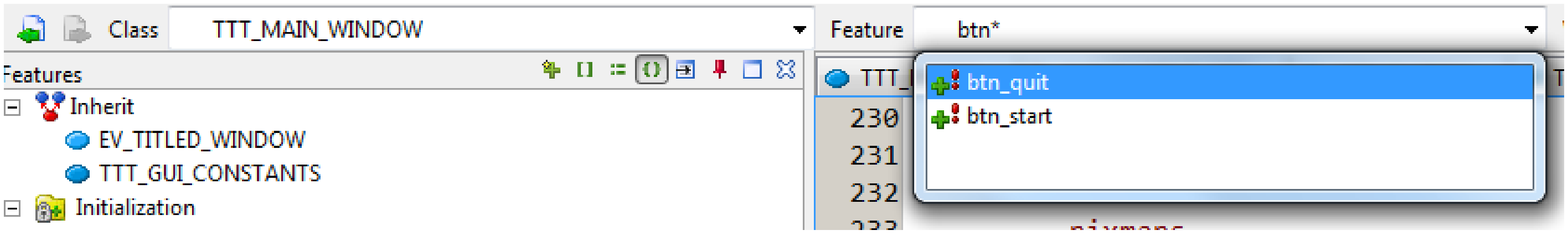


Editable views	
Basic text view	Default editor view, used for writing code
Non-editable views	
Clickable view	Reformatted representation of the code; more clickable elements (e.g. comments) than in basic text view
Flat view	Shows the flattened version of a class (e.g. all inherited features); this is the view used by the debugger
Contract View	Public interface of the class, incl. contracts
Interface view	Same as contract view but for the flat-version of the class.

more: http://docs.eiffel.com/book/eiffelstudio/class-views

Code Browsing – the Adressbar

- ***Adressbar*** can be used to quickly open classes or features



- If you only remember part of a class or feature name, use “ * ” in the search, e.g.
  - Search class: ***TTT_**** → all classes starting with TTT
  - Search feature: ***** → all features of the current class

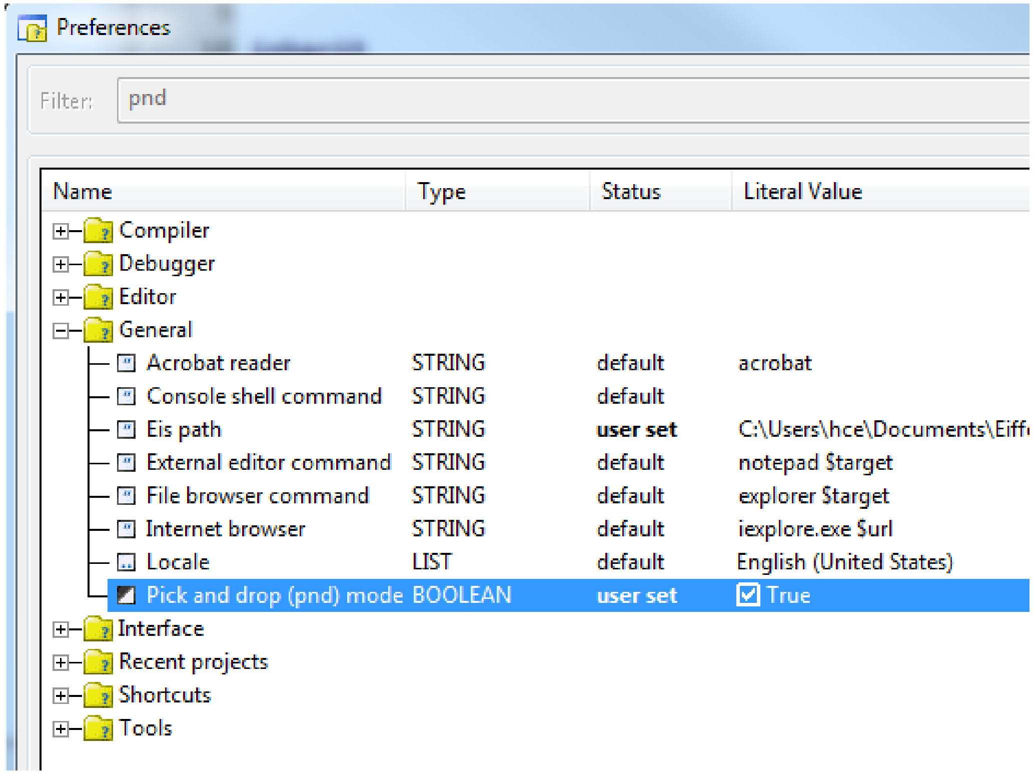
Tip for Pick & Drop

- Many ES tools work with ***Pick & Drop***
    - Tip: try to drop elements on various kinds of icons in ES

- ***Pick & Drop*** feels “slow” if you go through the right-click context menu

- Make right-click the default for Pick & Drop:

  Tools -> Preferences -> General.Pick and drop


Preferences

Filter: pnd

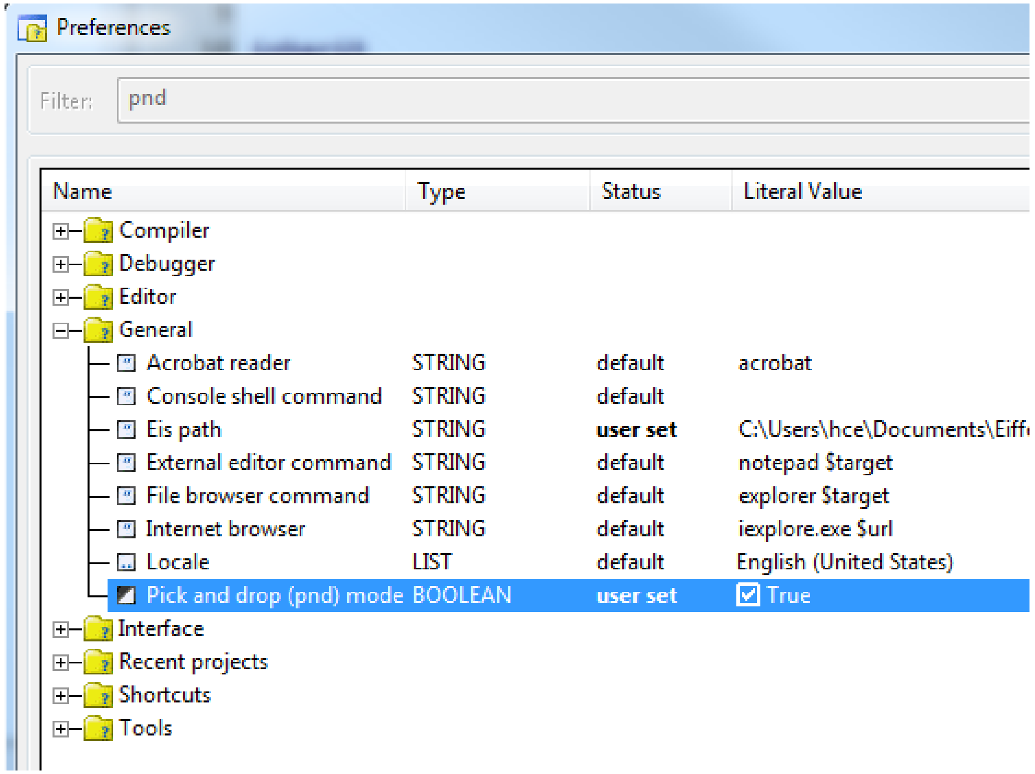
Name	Type	Status	Literal Value
⊞ Compiler			
⊞ Debugger			
⊞ Editor			
⊟ General			
Acrobat reader	STRING	default	acrobat
Console shell command	STRING	default	
Eis path	STRING	**user set**	C:\Users\hce\Documents\Eiff
External editor command	STRING	default	notepad $target
File browser command	STRING	default	explorer $target
Internet browser	STRING	default	iexplore.exe $url
Locale	LIST	default	English (United States)
Pick and drop (pnd) mode	BOOLEAN	**user set**	☑ True
⊞ Interface			
⊞ Recent projects			
⊞ Shortcuts			
⊞ Tools			

Tip for Pick & Drop

- Option 1:
  - Shift + right-click: Pick
  - Ctrl + right-click: Open element in a new tab in editor

- Option 2:
  - Make right-click the default for Pick & Drop:

    Tools -> Preferences -> General.Pick and drop



Preferences

Filter: pnd

Name	Type	Status	Literal Value
Compiler			
Debugger			
Editor			
General			
Acrobat reader	STRING	default	acrobat
Console shell command	STRING	default	
Eis path	STRING	**user set**	C:\Users\hce\Documents\Eiff
External editor command	STRING	default	notepad $target
File browser command	STRING	default	explorer $target
Internet browser	STRING	default	iexplore.exe $url
Locale	LIST	default	English (United States)
Pick and drop (pnd) mode	BOOLEAN	**user set**	✓ True
Interface			
Recent projects			
Shortcuts			
Tools			

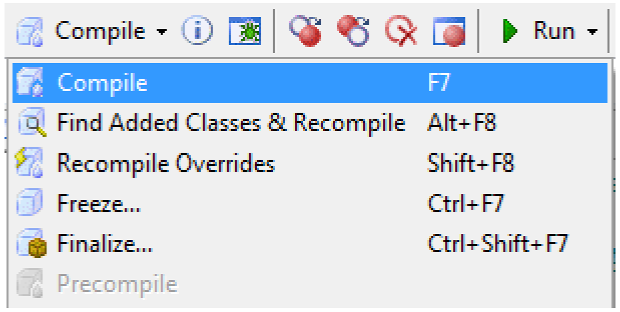
Compiler

- Compiling a system (F7)

- ***Melting***: Generates bytecode, not C code. Quick to generate but slowest execution. Use during development.

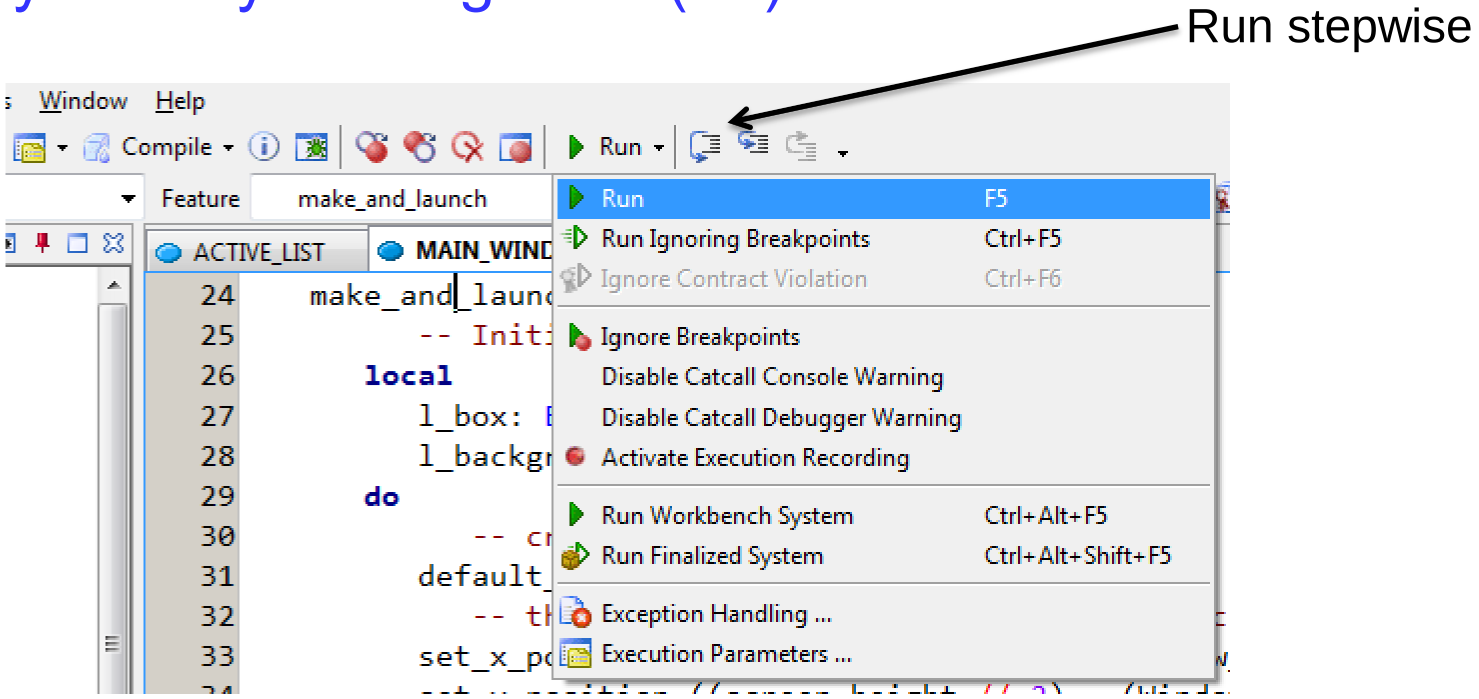
- ***Freezing***: Generates C code for the whole system. Compilation takes longer but system executes faster. Can still be debugged. Use during development.

- ***Finalizing***: Creates an executable production version. Finalization performs extensive time and space optimizations. Cannot be debugged.


Compile
Run
Compile F7
Find Added Classes & Recompile Alt+F8
Recompile Overrides Shift+F8
Freeze... Ctrl+F7
Finalize... Ctrl+Shift+F7
Precompile

Running a System

- Run a system by clicking "run" (F5)



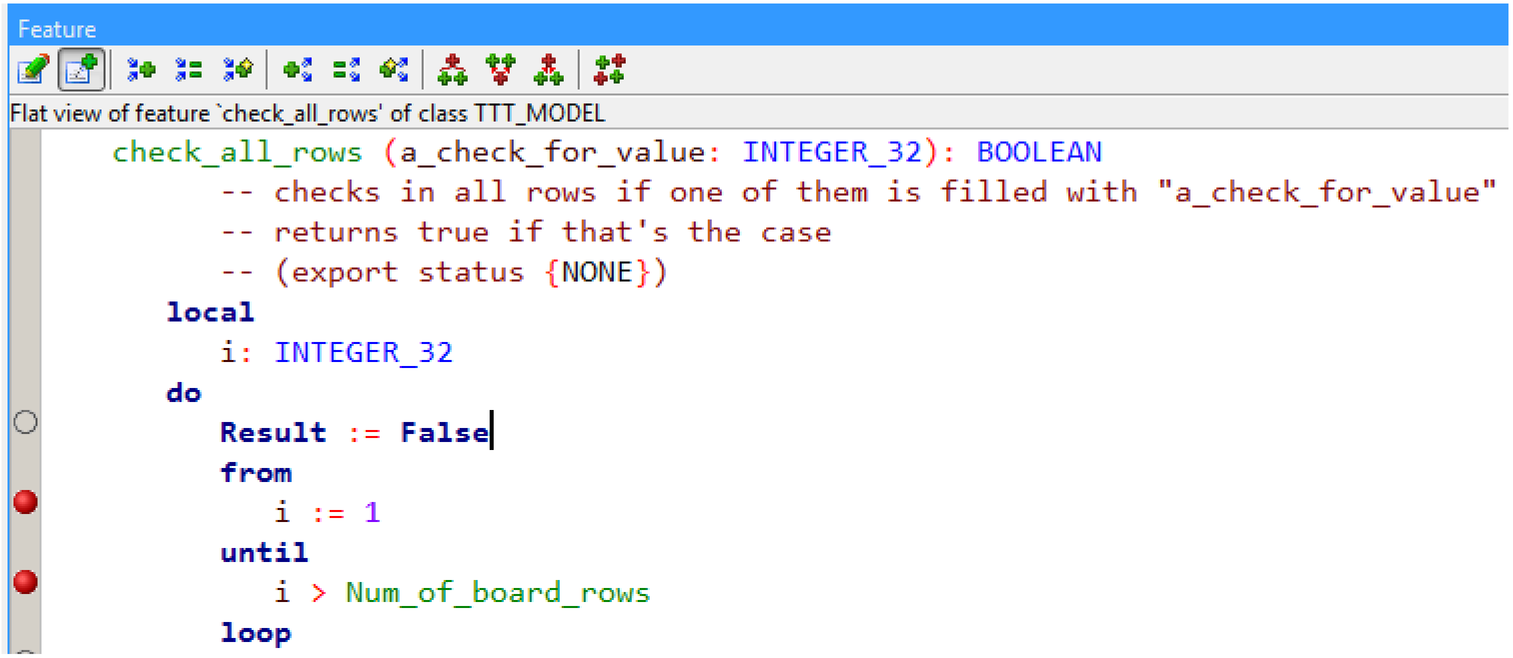
Run	F5
Run Ignoring Breakpoints	Ctrl+F5
Ignore Contract Violation	Ctrl+F6
Ignore Breakpoints	
Disable Catcall Console Warning	
Disable Catcall Debugger Warning	
Activate Execution Recording	
Run Workbench System	Ctrl+Alt+F5
Run Finalized System	Ctrl+Alt+Shift+F5
Exception Handling ...	
Execution Parameters ...	

```
make_and_launc
        -- Initi
    local
        l_box: E
        l_backgr
    do
            -- cr
        default_
            -- th
        set_x_po
```

- Switches to "execution mode"
  - Shows more debugging related tools
  - Shows controls for system execution (stop, pause, etc.)

Debugger

- Debugging works for *melted* and frozen *systems* (not for finalized ones)
- ***Breakpoints*** can only be added using a ***flat view***
  - One way: switch editor view to flat view
  - Quicker: Pick & Drop feature into Feature tool



Flat view of feature `check_all_rows' of class TTT_MODEL

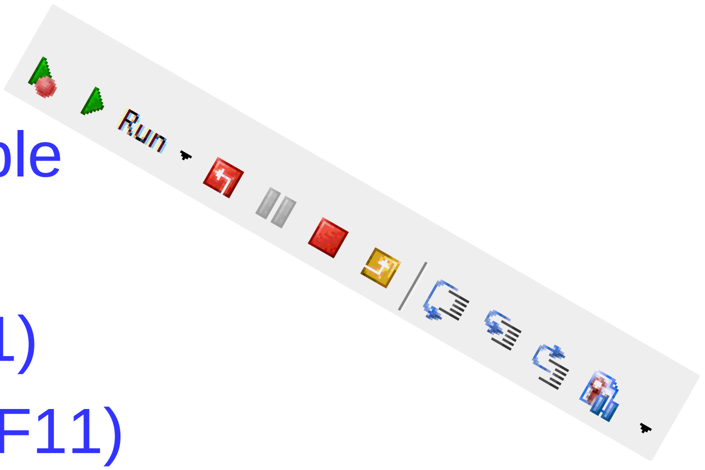
```
    check_all_rows (a_check_for_value: INTEGER_32): BOOLEAN
            -- checks in all rows if one of them is filled with "a_check_for_value"
            -- returns true if that's the case
            -- (export status {NONE})
        local
            i: INTEGER_32
        do
            Result := False
            from
                i := 1
            until
                i > Num_of_board_rows
            loop
```

Debugger

- To start the debugger simply hit “Run”
  (no distinction like in Eclipse)

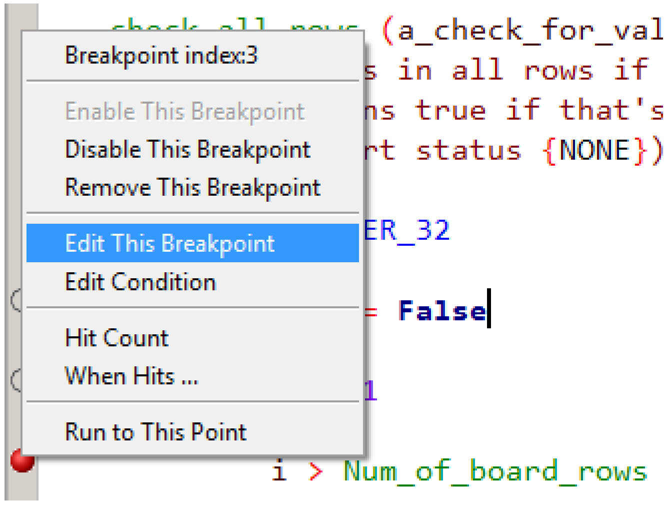
- Usual debugger tools are available during debugging
  - Call Stack
  - Expressions
  - Switching between threads

- Usual debugger controls are available
  - One step at a time (F10)
  - Step into a routine value (F11)
  - Step out of a routine (Shift + F11)



Debugger

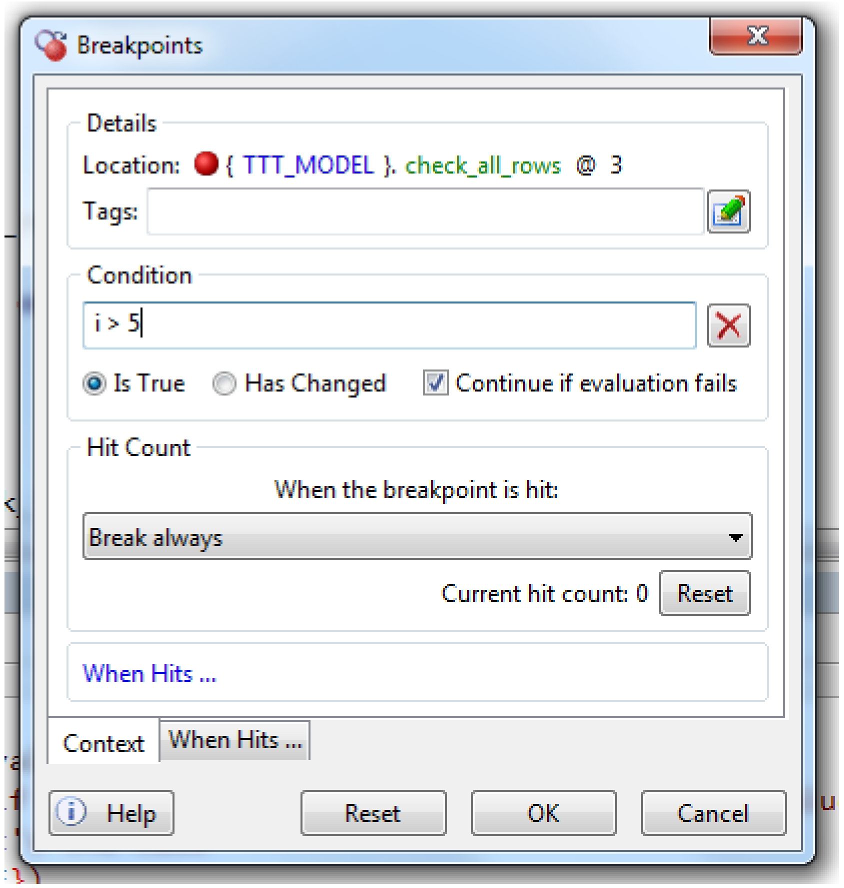
- Often useful: ***conditional breakpoints***
  - Execution will only be stop under certain condition



Breakpoint index:3

Enable This Breakpoint
Disable This Breakpoint
Remove This Breakpoint
Edit This Breakpoint
Edit Condition
Hit Count
When Hits ...
Run to This Point

```
check_all_rows (a_check_for_val
              s in all rows if
              ns true if that's
              rt status {NONE})
              ER_32
              = False
              1
        i > Num_of_board_rows
```



Details
Location: { TTT_MODEL }. check_all_rows @ 3
Tags:

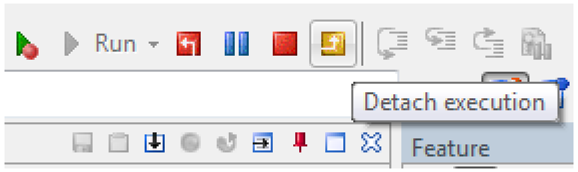
Condition
i > 5
[✓ Is True] Has Changed [✓ Continue if evaluation fails]

Hit Count
When the breakpoint is hit:
Break always
Current hit count: 0

Tip for debugging a client/server system

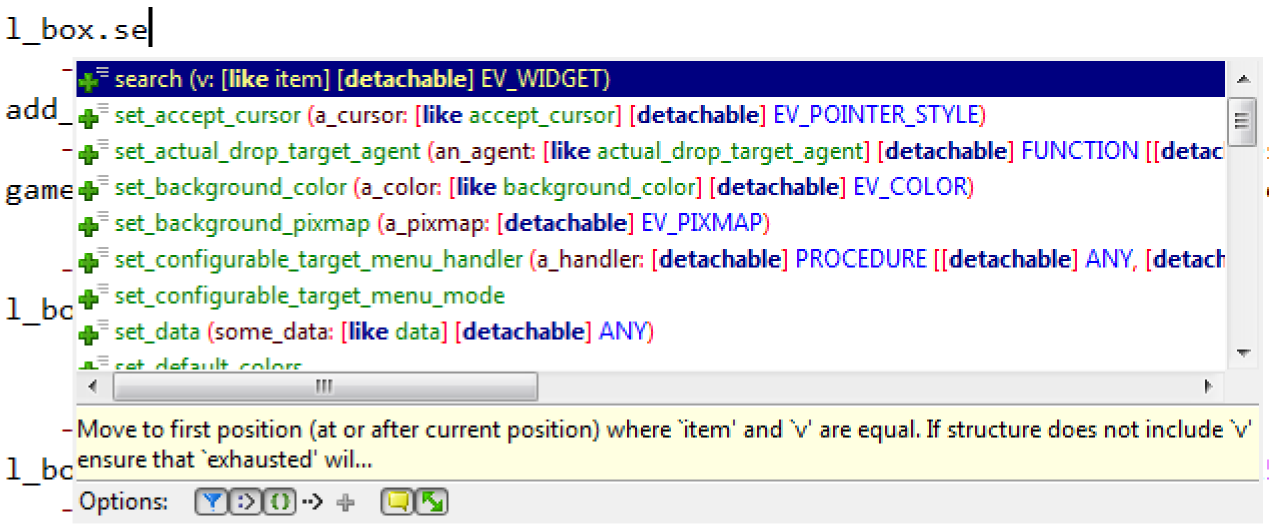
- Goal: run server and client on same machine

- Rather than using command line, you can do:
  - Run the server
  - ***Detach*** the server instance
  - ES returns to “edit mode”
  - Run the client



Auto-completion

- ES has auto-completion



```
l_box.se
```

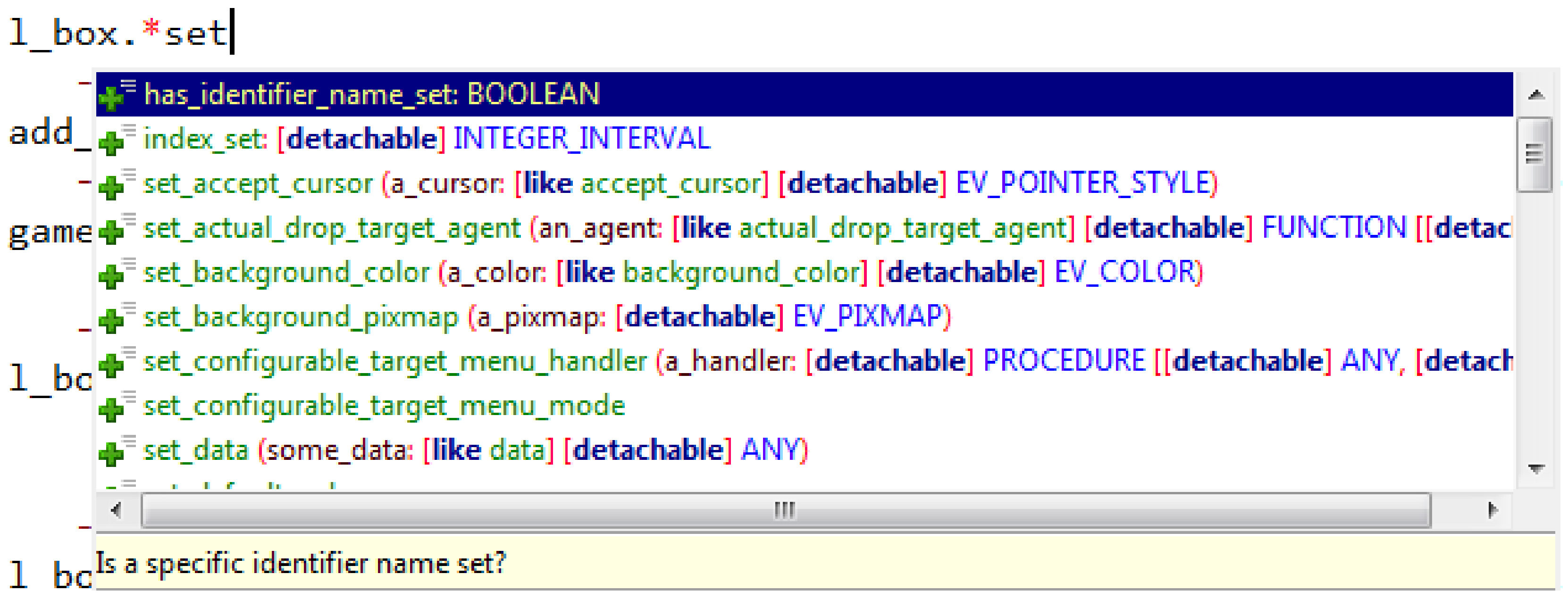
search (v: [like item] [detachable] EV_WIDGET)
set_accept_cursor (a_cursor: [like accept_cursor] [detachable] EV_POINTER_STYLE)
set_actual_drop_target_agent (an_agent: [like actual_drop_target_agent] [detachable] FUNCTION [[detac
set_background_color (a_color: [like background_color] [detachable] EV_COLOR)
set_background_pixmap (a_pixmap: [detachable] EV_PIXMAP)
set_configurable_target_menu_handler (a_handler: [detachable] PROCEDURE [[detachable] ANY, [detach
set_configurable_target_menu_mode
set_data (some_data: [like data] [detachable] ANY)
set_default_colors

Move to first position (at or after current position) where `item' and `v' are equal. If structure does not include `v' ensure that `exhausted' wil...

- Auto-completion knows (only) about compiled code
- If it does not work (as you would expect), try the following
  - Try to compile the system
  - Close and reopen the file in the editor
  - Do a clean compile

Auto-completion

- “ * ” can also be used in auto-completion
- Example:
  - Find all calls containing “set”
  - Use ***my_target_name.*set***



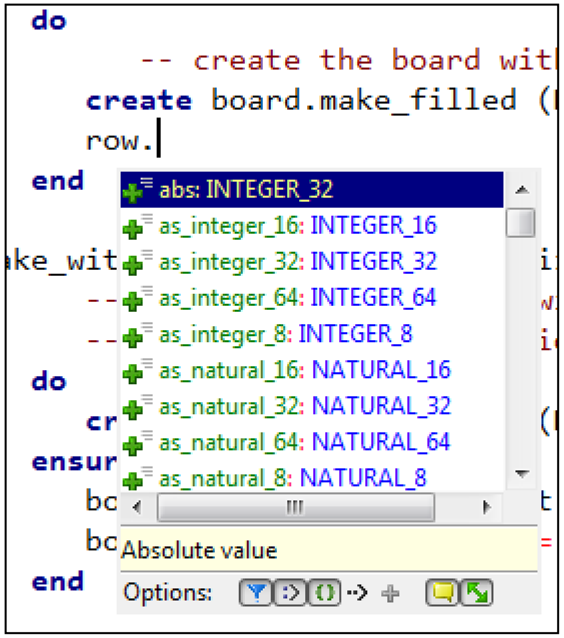
```
l_box.*set
has_identifier_name_set: BOOLEAN
index_set: [detachable] INTEGER_INTERVAL
set_accept_cursor (a_cursor: [like accept_cursor] [detachable] EV_POINTER_STYLE)
set_actual_drop_target_agent (an_agent: [like actual_drop_target_agent] [detachable] FUNCTION [[detac
set_background_color (a_color: [like background_color] [detachable] EV_COLOR)
set_background_pixmap (a_pixmap: [detachable] EV_PIXMAP)
set_configurable_target_menu_handler (a_handler: [detachable] PROCEDURE [[detachable] ANY, [detach
set_configurable_target_menu_mode
set_data (some_data: [like data] [detachable] ANY)
Is a specific identifier name set?
```

Auto-completion – Word vs. Class

We have two types of completion

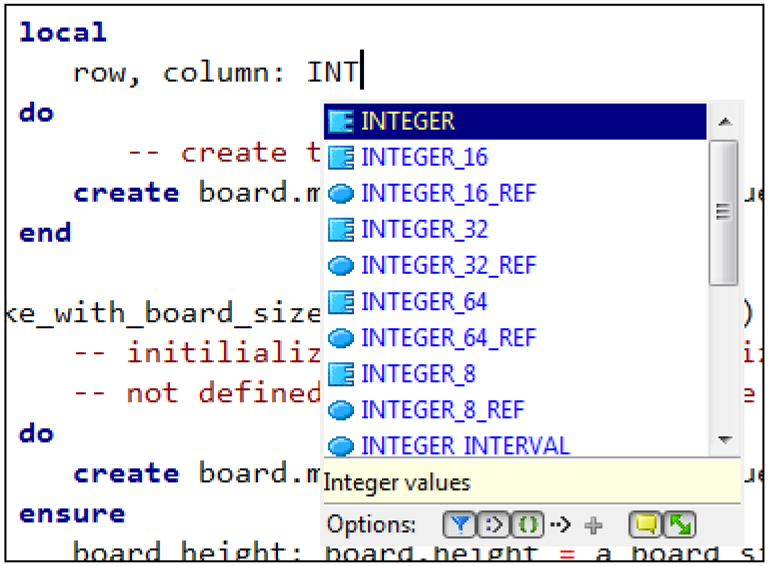
1. Word-completion (Ctrl + Space)

2. Class name completion (Ctrl + Shift + Space)



```
do
        -- create the board wit
    create board.make_filled (
    row.
end
```

```
abs: INTEGER_32
as_integer_16: INTEGER_16
as_integer_32: INTEGER_32
as_integer_64: INTEGER_64
as_integer_8: INTEGER_8
as_natural_16: NATURAL_16
as_natural_32: NATURAL_32
as_natural_64: NATURAL_64
as_natural_8: NATURAL_8
Absolute value
Options:
```



```
local
    row, column: INT
do
        -- create t
    create board.m
end

ke_with_board_size
    -- initializ
    -- not defined
do
    create board.m
ensure
    board_height: board.height = a_board_si
```

```
INTEGER
INTEGER_16
INTEGER_16_REF
INTEGER_32
INTEGER_32_REF
INTEGER_64
INTEGER_64_REF
INTEGER_8
INTEGER_8_REF
INTEGER_INTERVAL
Integer values
Options:
```

Code Templates

ES comes with a number of code templates

- write a keyword
- hit enter
- subsequent keywords are filled in automatically

Examples

- **do .. end**
- **from ... until … loop … end**
- **across … as … loop … end**

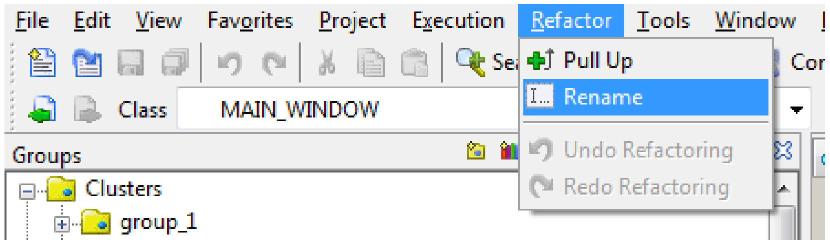
Special case → across loop + hitting space rather than enter

- **across … as … all … end**

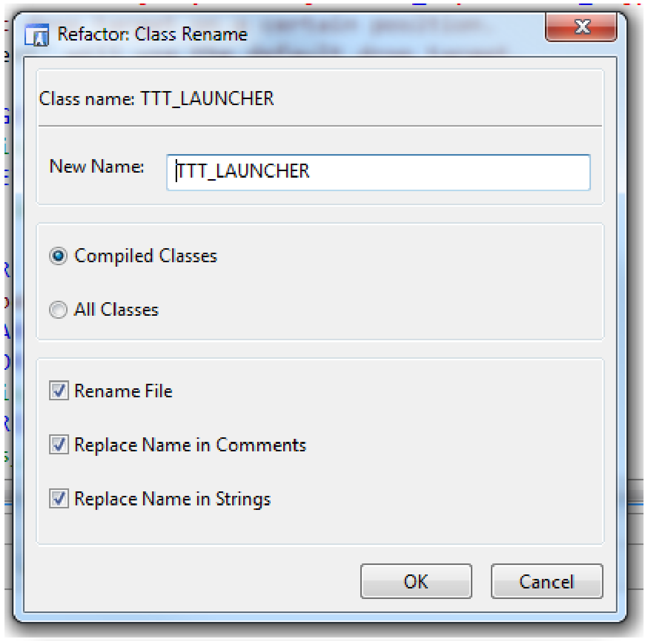
Refactoring Tools

- EiffelStudio only supports two refactorings:
    - Renaming
    - Pull Up routine

- Works only on compiling system



rename class



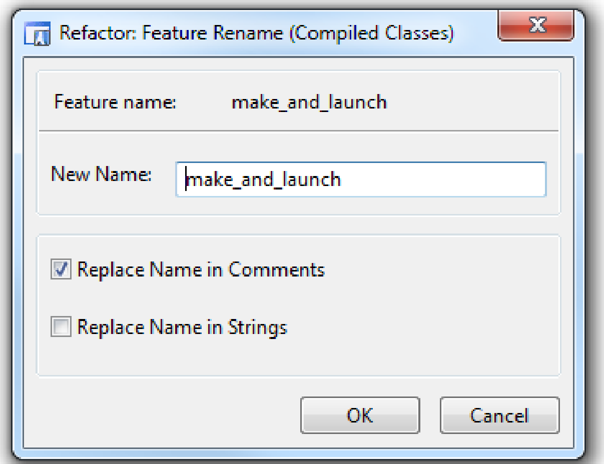
Class name: TTT_LAUNCHER

New Name: TTT_LAUNCHER

[● Compiled Classes] ○ All Classes

[✓] Rename File
[✓] Replace Name in Comments
[✓] Replace Name in Strings

rename feature



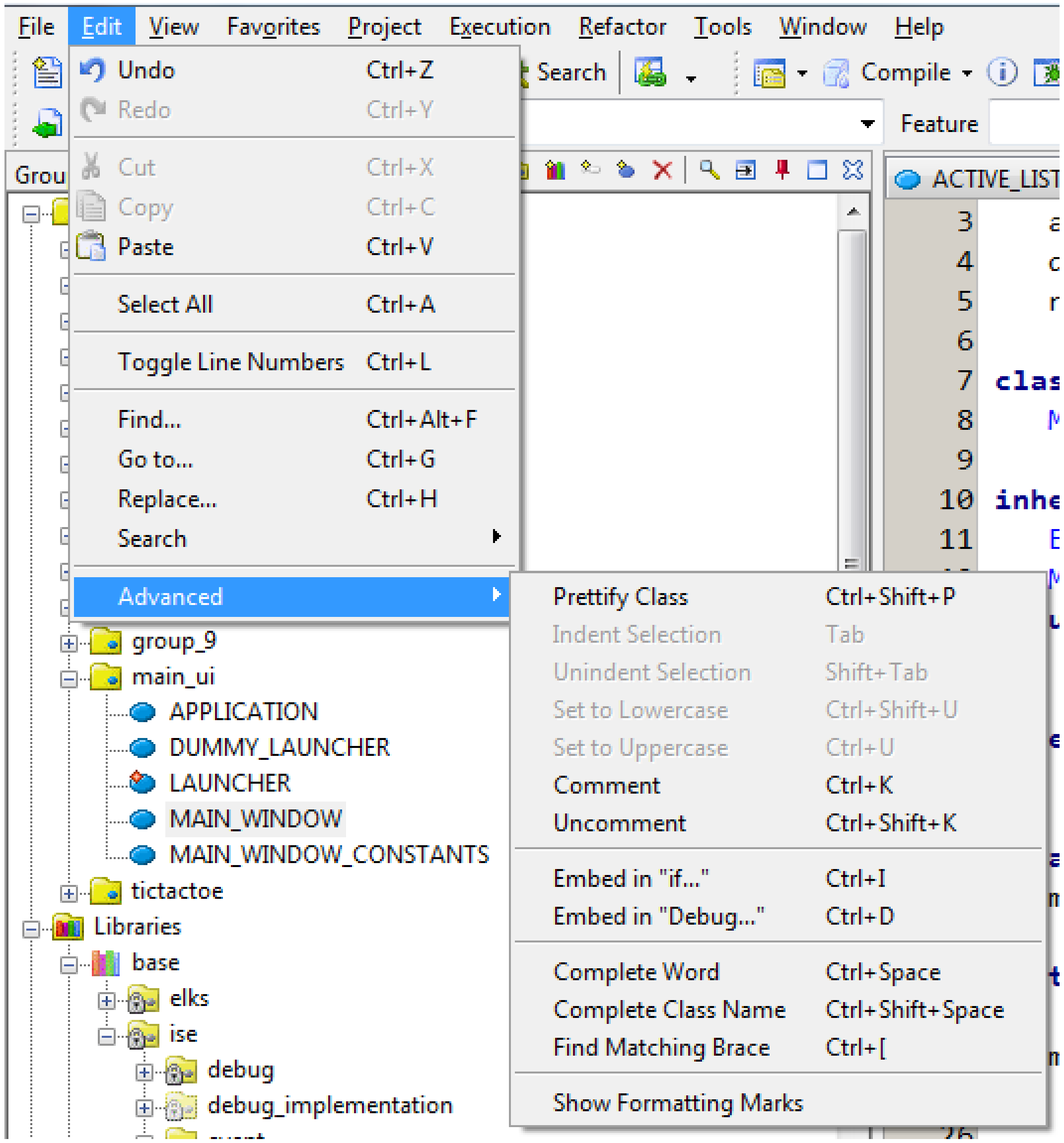
Feature name: make_and_launch

New Name: make_and_launch

[✓] Replace Name in Comments
[ ] Replace Name in Strings

Other useful stuff

- Take a look at menu
  - Edit
  - Edit → Advanced
- Make use of
  - Line numbers
  - Pretty print
  - Commenting
  - …
- Learn some of keyboard shortcuts ☺



Things we ignore for the moment

- Project settings
  - Shared with all other teams
  - Thus you should not modify them
- Profiler: http://docs.eiffel.com/book/eiffelstudio/profiling
- Record Replay: http://docs.eiffel.com/book/eiffelstudio/execution-record-and-replay

Further Resources

- Official EiffelSoftware websites:
  - http://www.eiffel.com/
  - http://dev.eiffel.com/
  - http://docs.eiffel.com/

- Have a look in the DOSE wiki

- Make use of the Eiffel Mailing list:
  - http://tech.groups.yahoo.com/group/eiffel_software/
  - Search the archive for answers
  - Feel free to post any kind of (Eiffel-related) question

THE END