## ADT Specification

NAME

Text Editor – an ADT representing a text editor for (insert functions, brief)

SETS

T the set of Text Editors, { ( [C] x [C] x [C] x [C] ) }

C the set of Characters, { a..z ∪ A..Z ∪ 0..9 ∪ ., ,,,:,;,’,”,!,?,(,),[,] }

N the set of Natural Numbers, { ∀*n.n*∈Ζ∧*n*≥0∧*n*≤1024 }

E the set of Error Messages, { [C] }

F the set of text files, { [C] }

SYNTAX

Create: ⊥ → T

Destroy: T → ⊥

Init: T → T

GetLeftOfCursor: T → [C]

SetLeftOfCursor: T x [C] → T

GetHighlighted: T → [C]

SetHighlighted: T x [C] → T

GetRightOfCursor: T → [C]

SetRightOfCursor: T x [C] → T

GetClipboard: T → [C]

SetClipboard: T x [C] → T

AddToString: T x [C] → T

DeleteLeftOfCursor: T → T

DeleteRightOfCursor: T → T

DeleteWord: T → T

MoveCursorLeft: T → T

MoveCursorRight: T → T

MoveCursorWordLeft: T → T

MoveCursorWordRight: T → T

MoveCursorBeginning: T → T

MoveCursorEnd: T → T

CharacterCount: T → N

HighlightLeft: T → T

HighlightRight: T → T

HighlightWordLeft: T → T

HighlightWordRight: T → T

HighlightWord: T → T

HighlightAllLeft: T → T

HighlightAllRight: T → T

Copy: T → T

Cut: T → T

DeleteHighlighted: T → T

Paste: T → T

ClearClipboard: T → T

LoadFile: F → T

SaveFile: T x [C] → F

SEMANTICS

∀ l.l ∈ [C], ∀ h.h ∈ [C], ∀ a.a ∈ [C], ∀ b.b ∈ [C], ∀ u.u ∈ [C]

pre-Create() :: true

post-Create(t; r) :: r = ( [], [], [], [] )

pre-Destroy(t) :: true

post-Destroy(t; r) :: r = ⊥

pre-Init(t) :: true

post-Init ((\_,\_,\_,\_); r) :: r = ( [], [], [], [] )

pre-GetLeftOfCursor(t) :: true

post-GetLeftOfCursor((l,\_,\_,\_); r) :: r = l

pre-SetLeftOfCursor(t,[c]) :: true

post-SetLeftOfCursor((\_,h,a,b),u; r) :: r = ( u, h, a, b )

pre-GetHighlighted(t) :: true

post-GetHighlighted((\_,h,\_,\_); r) :: r = h

pre-SetHighlighted(t,[c]) :: true

post-SetHighlighted((l,\_,a,b),u; r) :: r = ( l, u, a, b )

pre-GetRightOfText(t) :: true

post-GetRightOfText((\_,\_,a,\_); r) :: r = a

pre-SetRightOfText(t,[c]) :: true

post-SetRightOfText((l,h,\_,b),u; r) :: r = ( l, h, u, b )

pre-GetClipboard(t) :: true

post-GetClipboard((\_,\_,\_,b); r) :: r = b

pre-SetClipboard(t,[c]) :: true

post-SetClipboard((l,h,a,\_),u; r) :: r = ( l, h, a, u )

pre-AddToString(t,[c]) :: true

post-AddToString((l,h,a,b),u; r) :: r = ( l ∧∧ u, h, a, b )

pre-DeleteLeftOfCursor(t) :: true

post-DeleteLeftOfCursor((l,h,a,b); r) :: r = ( (init l), h, a, b )

pre-DeleteRightOfCursor(t) :: true

post-DeleteRightOfCursor((l,h,a,b); r) :: r = ( l, h, (tail a), b )

pre-DeleteWord(t) :: true

post-DeleteWord((l,h,a,b); r) ::

r = DeleteWordRec( l, h, a, b )

Where

DeleteWordRec( l, h, a, b ) = ( l, h, a, b )

DeleteWordRec( l, h, a, b ) =

WHEN l /= [] ∧ (last l) /= “ “

THEN DeleteWordRec( (init l), h, a, b )

ELSE WHEN a /= [] ∧ (head a) /= “ “

THEN DeleteWordRec( l, h, (tail a), b )

ELSE ( l, h, a, b )

pre-MoveCursorLeft(t) :: true

post-MoveCursorLeft((l,h,a,b); r) ::

r = MoveCursorLeft( l, h, a, b )

Where

MoveCursorLeft( l, h, a, b ) =

WHEN l /= []

THEN ( (init l), [], (last l) ∧∧ h ∧∧ a, b )

ELSE ( l, h, a, b )

pre-MoveCursorRight(t) :: true

post-MoveCursorRight((l,h,a,b); r) ::

r = MoveCursorRight( l, h, a, b )

Where

MoveCursorRight( l, h, a, b ) =

WHEN a /= []

THEN ( l ∧∧ h ∧∧ (head a), [], (tail a), b )

ELSE ( l, h, a, b )

pre-MoveCursorWordLeft(t) :: true

post-MoveCursorWordLeft((l,h,a,b); r) ::

r = MoveWordLeftRec( l, h, a, b )

Where

MoveWordLeftRec( l, h, a, b ) = ( l, h, a, b )

MoveWordLeftRec( l, h, a, b ) =

WHEN l /= [] ∧ (last l) /= “ “

THEN MoveWordLeftRec( (init l), [], (last l) ∧∧ h ∧∧ a, b )

ELSE ( l, h, a, b )

pre-MoveCursorWordRight(t) :: true

post-MoveCursorWordRight((l,h,a,b); r) ::

r = MoveWordRightRec( l, h, a, b )

Where

MoveWordRightRec( l, h, a, b ) = ( l, h, a, b )

MoveWordRightRec( l, h, a, b ) =

WHEN a /= [] ∧ (head a) /= “ “

THEN MoveWordRightRec( l ∧∧ h ∧∧ (head a), [], (tail a), b )

ELSE ( l, h, a, b )

pre-MoveCursorBeginning(t) :: true

post-MoveCursorBeginning((l,h,a,b); r) :: r = ( [], [], l ∧∧ h ∧∧ a, b )

pre-MoveCursorEnd(t) :: true

post-MoveCursorEnd((l,h,a,b); r) :: r = ( l ∧∧ h ∧∧ a, [], [], b )

pre-CharacterCount(t) :: true

post-CharacterCount((l,h,a,b); r) :: r = n( length(l ∧∧ h ∧∧ a) )

pre-HighlightLeft(t) :: true

post-HighlightLeft((l,h,a,b); r) :: r = ( (init l), (last l) ∧∧ h, a, b )

pre-HighlightRight(t) :: true

post-HighlightRight((l,h,a,b); r) :: r = ( l, h ∧∧ (head a), (tail a), b )

pre-HighlightWordLeft(t) :: true

post-HighlightWordLeft((l,h,a,b); r) ::

r = HighlightWordLeftRec( l, h, a, b )

Where

HlWordLeftRec( l, h, a, b ) = ( l, h, a, b )

HlWordLeftRec( l, h, a, b ) =

WHEN l /= [] ∧ (last l) /= “ “

THEN HlWordLeftRec( (init l), (last l) ∧∧ h, a, b )

ELSE ( l, h, a, b )

pre-HighlightWordRight(t) :: true

post-HighlightWordRight((l,h,a,b); r) ::

r = HlWordRightRec( l, h, a, b )

Where

HlWordRightRec( l, h, a, b ) = ( l, h, a, b )

HlWordRightRec( l, h, a, b ) =

WHEN a /= [] ∧ (head a) /= “ “

THEN HlWordRightRec( l, h ∧∧ (head a), (tail a), b )

ELSE ( l, h, a, b )

pre-HighlightWord(t) :: true

post-HighlightWord((l,h,a,b); r) ::

r = HlWordRec( l, h, a, b )

Where

HlWordRec( l, h, a, b ) = ( l, h, a, b )

HlWordRec( l, h, a, b ) =

WHEN l /= [] ∧ (last l) /= “ “

THEN HlWordRec( (init l), (last l) ∧∧ h, a, b )

ELSE WHEN a /= [] ∧ (head a) /= “ “

THEN HlWordRec( l, h ∧∧ (head a), (tail a), b )

ELSE ( l, h, a, b )

pre-HighlightAllLeft(t) :: true

post-HighlightAllLeft((l,h,a,b); r) :: r = ( [], l ∧∧ h, a, b )

pre-HighlightAllRight(t) :: true

post-HighlightAllRight((l,h,a,b); r) :: r = ( l, h ∧∧ a, [], b )

pre-Copy(t) :: true

post-Copy((l,h,a,b); r) :: r = ( l, h, a, h )

pre-Cut(t) :: true

post-Cut((l,h,a,b); r) :: r = ( l, [], a, h )

pre-DeleteHighlighted(t) :: true

post-DeleteHighlighted((l,h,a,b); r) :: r = ( l, [], a, b )

pre-Paste(t) :: true

post-Paste((l,h,a,b); r) :: r = ( l ∧∧ b, [], a, b )

pre-ClearClipboard(t) :: true

post-ClearClipboard((l,h,a,b); r) :: r = ( l, h, a, [] )

pre-LoadFile(f) :: true

post-LoadFile(t,f; r) :: r = ( f, [], [], [] )

pre-SaveFile(t) :: true

post-SaveFile((l,h,a,\_),f; r) :: r = ( l ∧∧ h ∧∧ a )

AUTHOR

Cameron Wragg, [15595673@students.lincoln.ac.uk](mailto:15595673@students.lincoln.ac.uk), April 2018