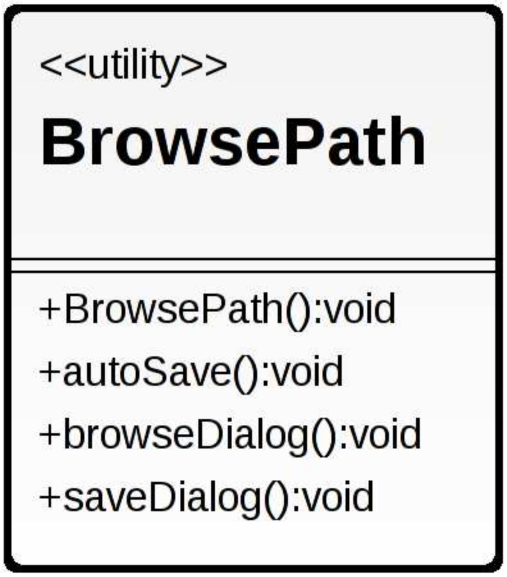
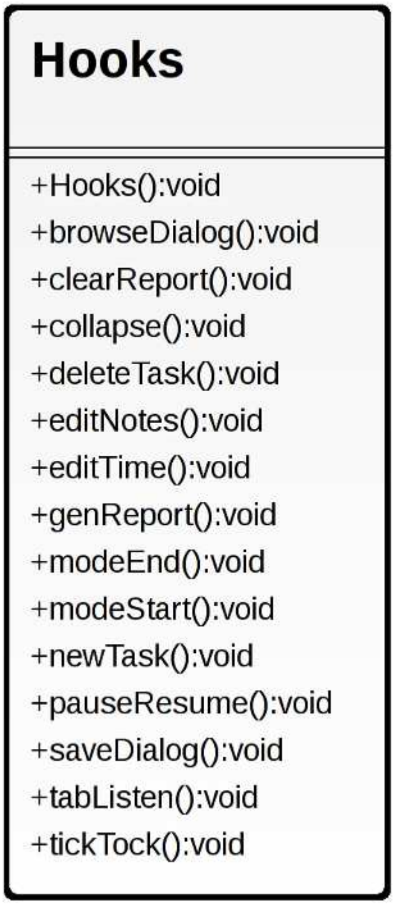


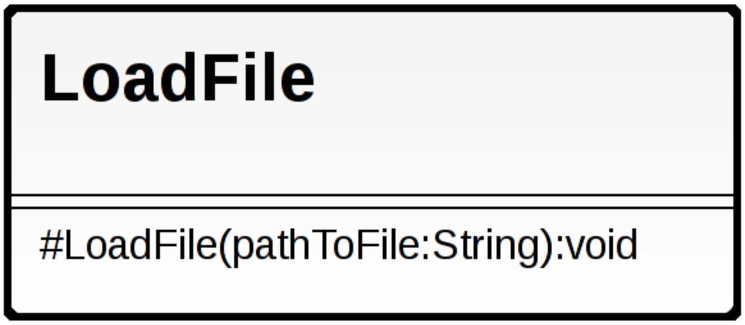
	BrowsePath	Native file selection handlers
	browseDialog	Loads file open selection in new window
	saveDialog	Loads file save selection in new window

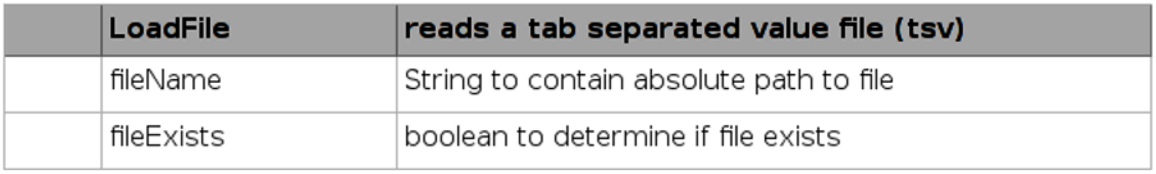


Hooks

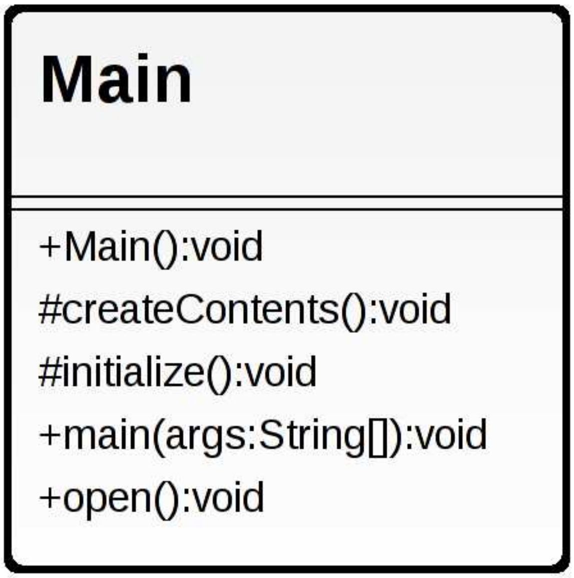
+Hooks():void
+browseDialog():void
+clearReport():void
+collapse():void
+deleteTask():void
+editNotes():void
+editTime():void
+genReport():void
+modeEnd():void
+modeStart():void
+newTask():void
+pauseResume():void
+saveDialog():void
+tabListen():void
+tickTock():void

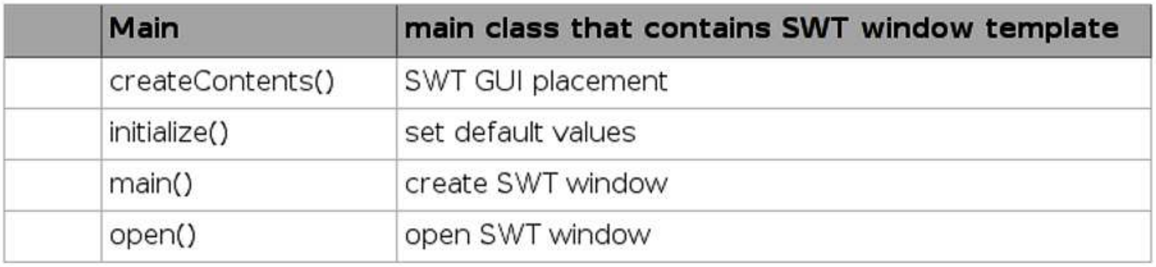
	Hooks	button listeners
	clearToggle	boolean to clear reports
	tickTock	edits pauseResume text, toggles StopWatch, adds elapsed to titlebar
	pauseResume	button triggers tickTock function
	newTask	saves current task, starts new task, changes to notes tab and resets values
	tabListen	monitors focused tab value
	genReport	enables report textbox, triggers newReport function
	clearReport	clears report textbox, disables report textbox and toggles clearToggle
	editNotes	grabs title and notes from select table row and inputs into notes tab
	editTime	creates in-line editor for elapsed time in selected row
	deleteTask	removes selected row from table and recent tasks (if applicable)
	modeStart	toggles report sort order by start time (default)
	modeEnd	toggles report sort order by end time
	collapse	toggles bottomPane, edits collapse button text



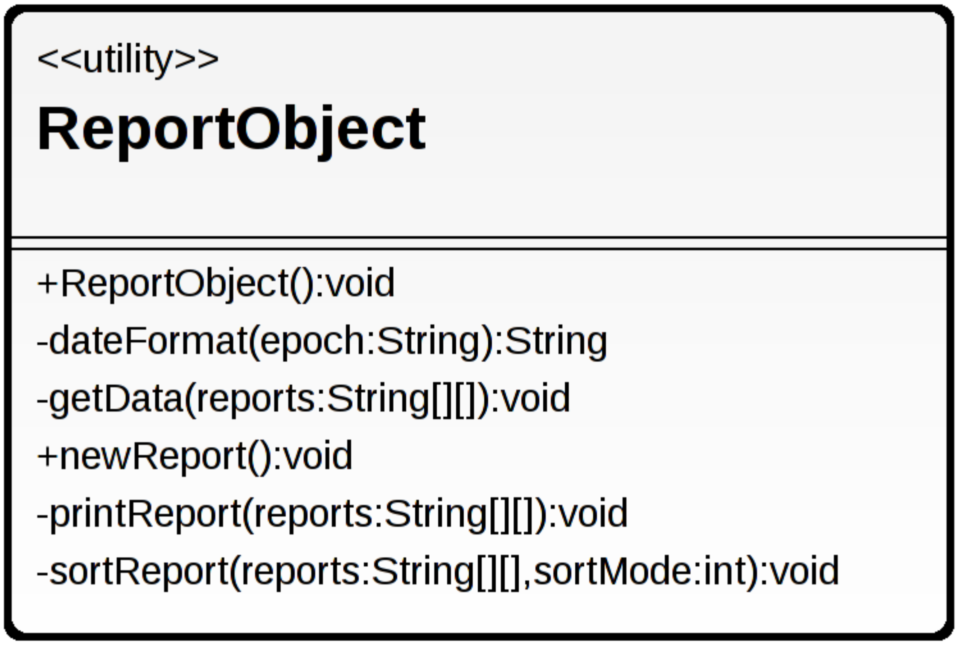


	LoadFile	reads a tab separated value file (tsv)
	fileName	String to contain absolute path to file
	fileExists	boolean to determine if file exists



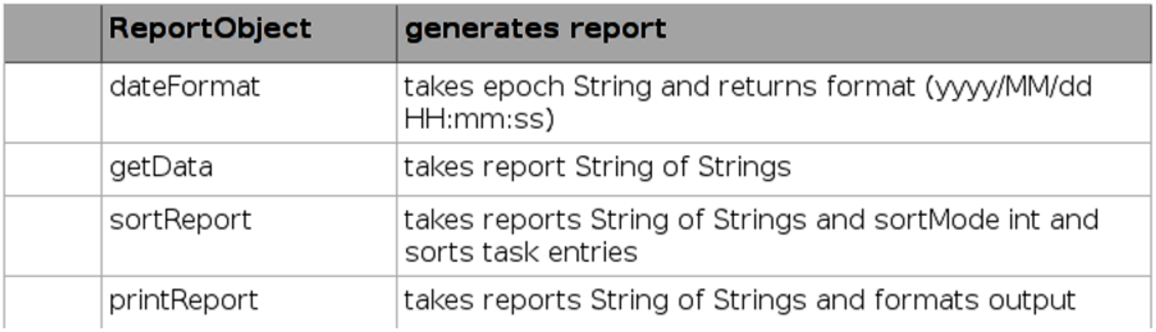


	Main	main class that contains SWT window template
	createContents()	SWT GUI placement
	initialize()	set default values
	main()	create SWT window
	open()	open SWT window

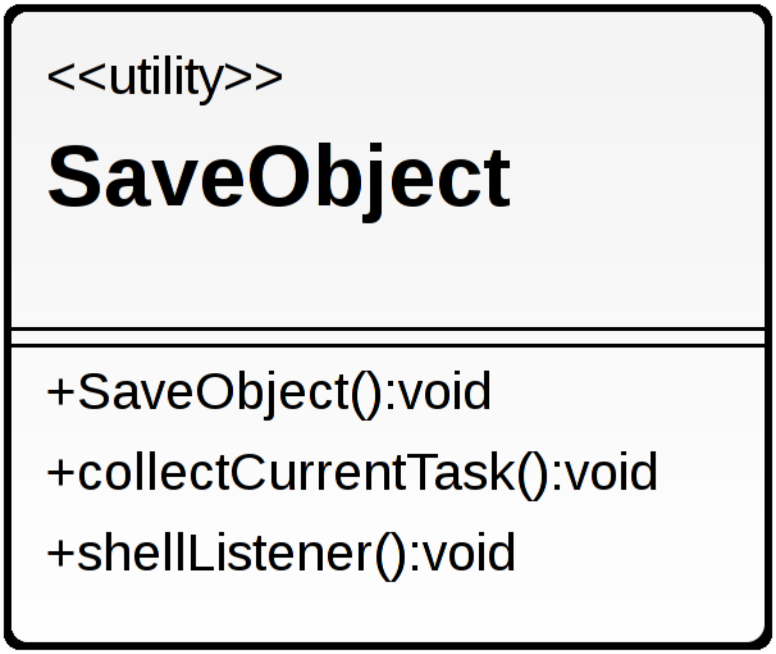

<<utility>>

ReportObject

+ReportObject():void
-dateFormat(epoch:String):String
-getData(reports:String[][]):void
+newReport():void
-printReport(reports:String[][]):void
-sortReport(reports:String[][],sortMode:int):void



	ReportObject	**generates report**
	dateFormat	takes epoch String and returns format (yyyy/MM/dd HH:mm:ss)
	getData	takes report String of Strings
	sortReport	takes reports String of Strings and sortMode int and sorts task entries
	printReport	takes reports String of Strings and formats output



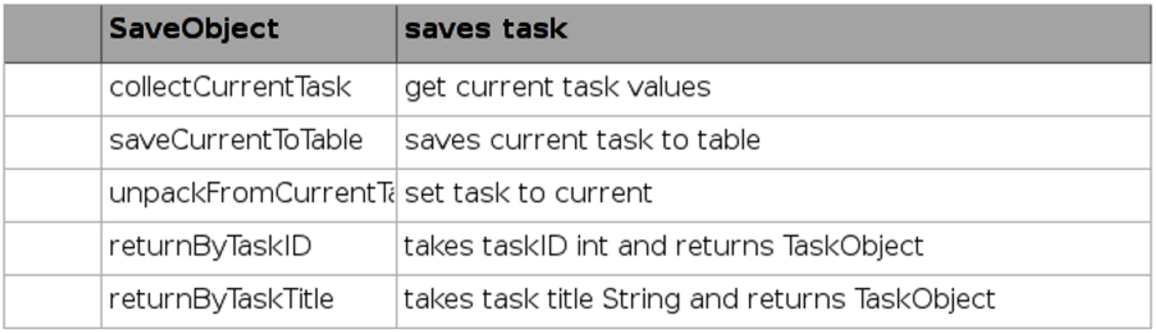
<<utility>>

SaveObject

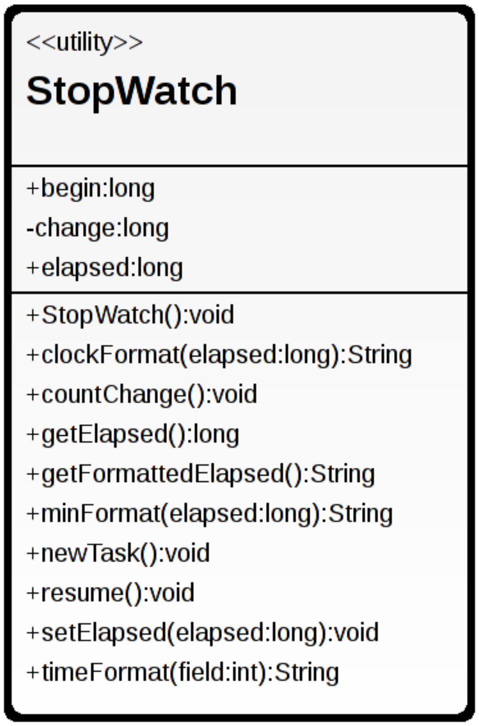
+SaveObject():void

+collectCurrentTask():void

+shellListener():void



	SaveObject	saves task
	collectCurrentTask	get current task values
	saveCurrentToTable	saves current task to table
	unpackFromCurrentTa	set task to current
	returnByTaskID	takes taskID int and returns TaskObject
	returnByTaskTitle	takes task title String and returns TaskObject



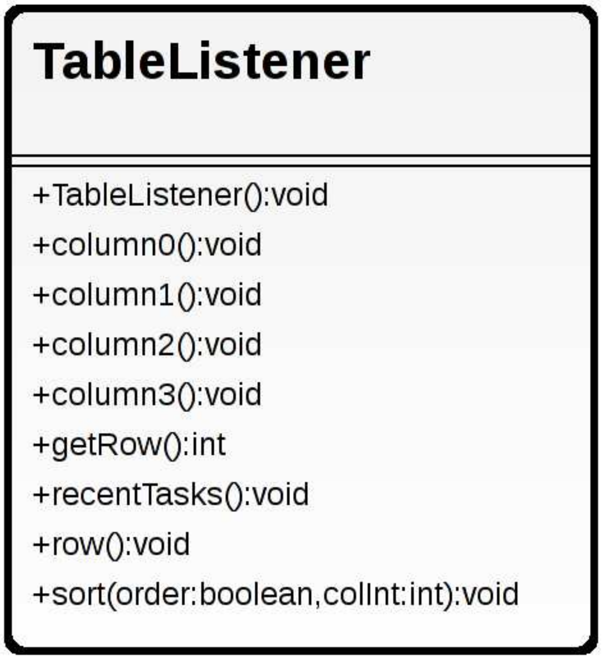
<<utility>>

StopWatch

+begin:long
-change:long
+elapsed:long

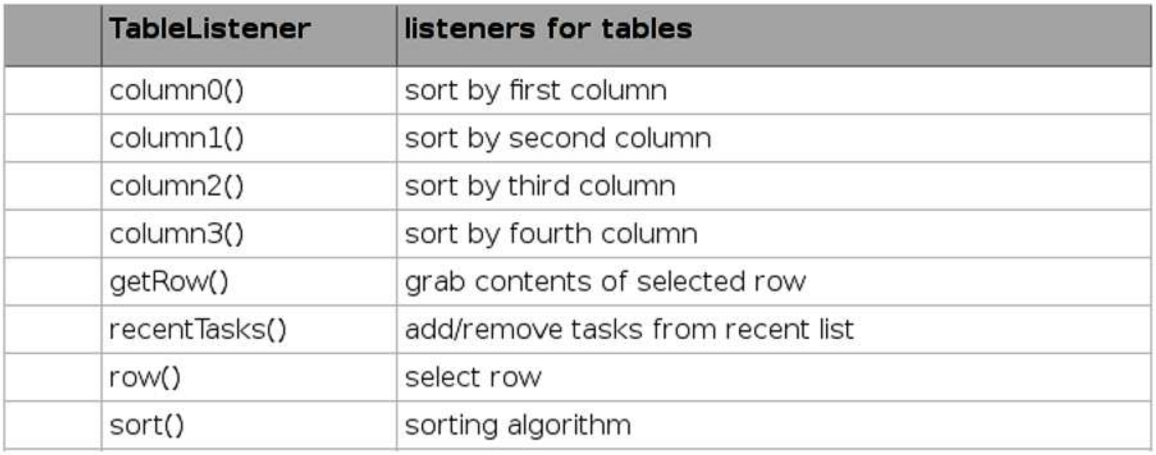
+StopWatch():void
+clockFormat(elapsed:long):String
+countChange():void
+getElapsed():long
+getFormattedElapsed():String
+minFormat(elapsed:long):String
+newTask():void
+resume():void
+setElapsed(elapsed:long):void
+timeFormat(field:int):String

	StopWatch	**timer**
	begin	long, initialized to current time
	change	long, intialized to 0
	elapsed	long, intialized to 0
	newTask	sets clock to 00:00:00
	resume	creates background thread that tracks elapsed time
	timeFormat	takes time int and returns two-digit String
	setElapsed	takes elapsed long and sets elapsed
	getElapsed	takes elapsed long and returns elapsed
	getFormattedElapsed	returns formatted elapsed time
	clockFormat	takes elapsed long and returns elapsed in HH:MM:SS.mm
	minFormat	takes elapsed long and returns HH:MM:SS
	countChange	sets title with elapsed every second



TableListener

+TableListener():void
+column0():void
+column1():void
+column2():void
+column3():void
+getRow():int
+recentTasks():void
+row():void
+sort(order:boolean,colInt:int):void



	TableListener	**listeners for tables**
	column0()	sort by first column
	column1()	sort by second column
	column2()	sort by third column
	column3()	sort by fourth column
	getRow()	grab contents of selected row
	recentTasks()	add/remove tasks from recent list
	row()	select row
	sort()	sorting algorithm