Parser

Routine	*Arguments*	*Returns*	*Function*
constructor	input file / stream	—	Opens the input file/stream, and gets ready to parse it.
`hasMoreLines`	—	boolean	Are there more lines in the input?
`advance`	—	—	Reads the next command from the input and makes it the *current command*. This method should be called only if `hasMoreLines` is true. Initially there is no current command.
`commandType`	—	`C_ARITHMETIC`, `C_PUSH`, `C_POP`, `C_LABEL`, `C_GOTO`, `C_IF`, `C_FUNCTION`, `C_RETURN`, `C_CALL` (constant)	Returns a constant representing the type of the current command. If the current command is an arithmetic-logical command, returns `C_ARITHMETIC`.
`arg1`	—	string	Returns the first argument of the current command. In the case of `C_ARITHMETIC`, the command itself (`add`, `sub`, etc.) is returned. Should not be called if the current command is `C_RETURN`.
`arg2`	—	int	Returns the second argument of the current command. Should be called only if the current command is `C_PUSH`, `C_POP`, `C_FUNCTION`, or `C_CALL`.

Same API as in project 7;

If your project 7 `Parser` did not handle the parsing of the VM commands:

`goto`, `if-goto`, `label`,

`call`, `function`, `return`,

add this parsing functionality now.

CodeWriter

Routine	*Arguments*	*Returns*	*Function*
constructor	output file / stream	—	Opens an output file / stream and gets ready to write into it. Writes the assembly instructions that effect the bootstrap code that starts the program's execution. This code must be placed at the beginning of the generated output file / stream.
`setFileName`	`fileName` (string)	—	Informs that the translation of a new VM file has started (called by the `VMTranslator`).
`writeArithmetic` (developed in project 7)	command (string)	—	Writes to the output file the assembly code that implements the given arithmetic-logical command.
`WritePushPop` (developed in project 7)	command (`C_PUSH` or `C_POP`), segment (string), `index` (int)	—	Writes to the output file the assembly code that implements the given `push` or `pop` command.

(API continues in the next slide)

CodeWriter

Routine	*Arguments*	*Returns*	*Function*
`writeLabel`	`label` (string)	—	Writes assembly code that effects the `label` command.
`writeGoto`	`label` (string)	—	Writes assembly code that effects the `goto` command.
`writeIf`	`label` (string)	—	Writes assembly code that effects the `if-goto` command.
`writeFunction`	`functionName` (string) `nVars` (int)	—	Writes assembly code that effects the `function` command.
`writeCall`	`functionName` (string) `nArgs` (int)	—	Writes assembly code that effects the `call` command.
`writeReturn`	—	—	Writes assembly code that effects the `return` command.
`close` (developed in project 7)	—	—	Closes the output file.

The generated assembly code must adhere to the symbol naming conventions.