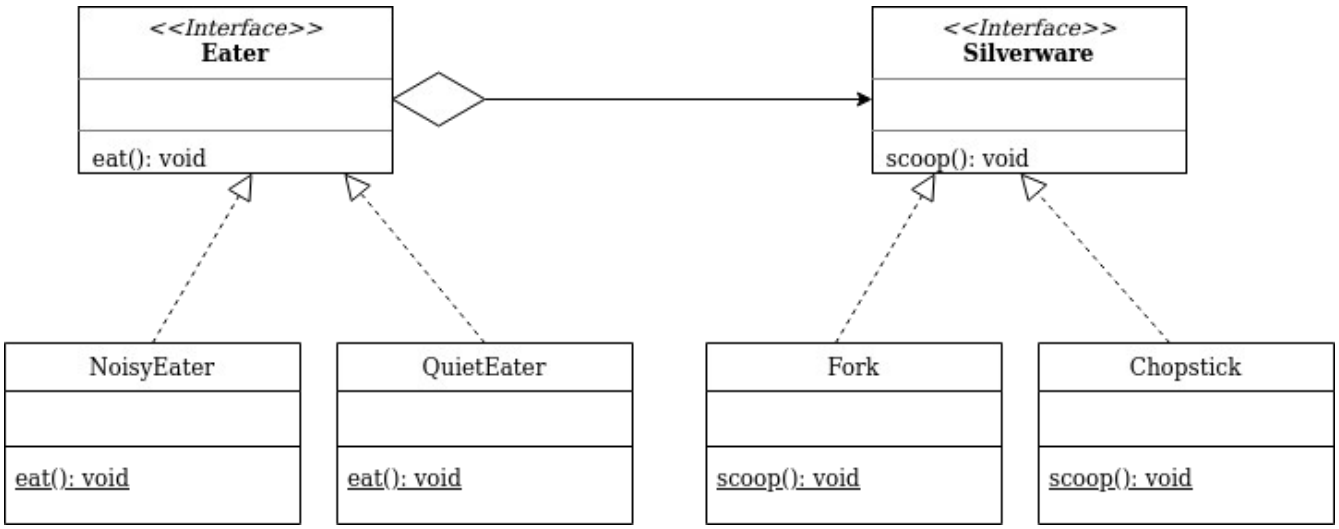
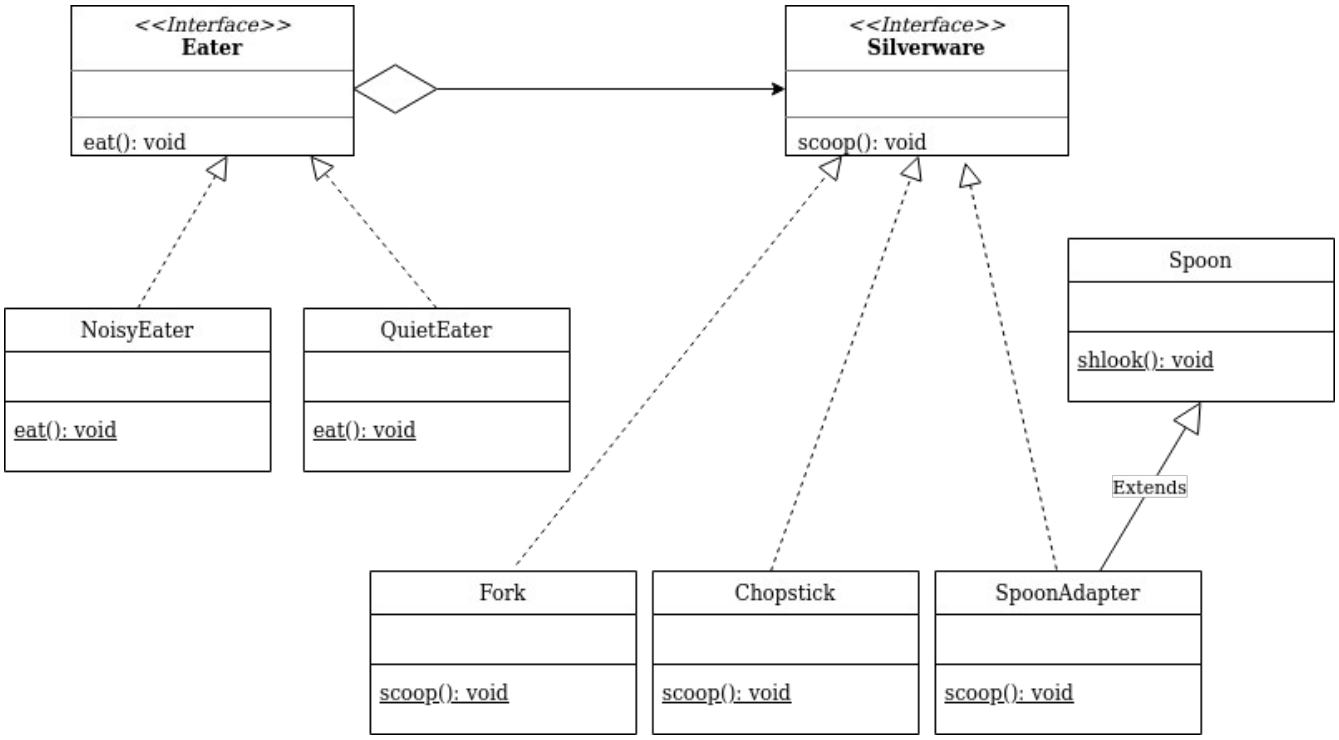
Tamar Harizy 209927128
Tali Cohen 329651871

Question1
Part A:



S: Each Module has a single responsibility : eat or scoop
O: Extendable via interface without modification (more eaters can be added, or more silverware, without modification of existing modules)
L: All instances implement the methods in their relevant interfaces, and knowledge of which instance is unnecessary
I: Eaters only exposed to Eater interface (with eat) and Silverware only exposed to silverware interface (with scoop)
D: No higher level dependant on a lower level, or abstract dependant on details

Part B:



S: Each Module has a single responsibility : eat or scoop

O: Extendable via interface without modification (more eaters can be added, or more silverware, without modification of existing modules as is shown by the addition of spoon)

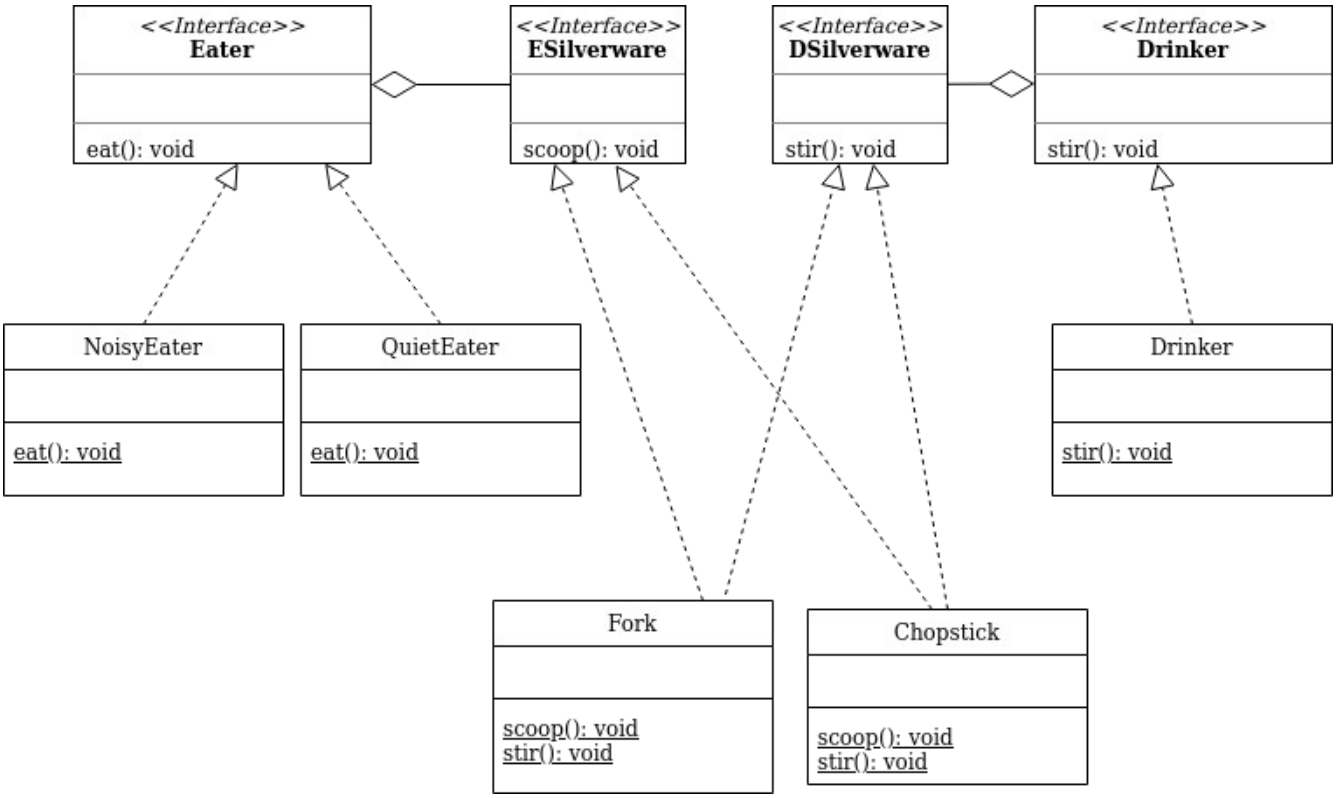
L: All instances implement the methods in their relevant interfaces, and knowledge of which instance is unnecessary

I: Eaters only exposed to Eater interface (with eat) and Silverware only exposed to silverware interface (with scoop) – via the spoon adaptor in the case of spoon

D: No higher level dependant on a lower level, or abstract dependant on details

(class adaption)

Part C:



S: Each Module has a single responsibility : eat/scoop/stir

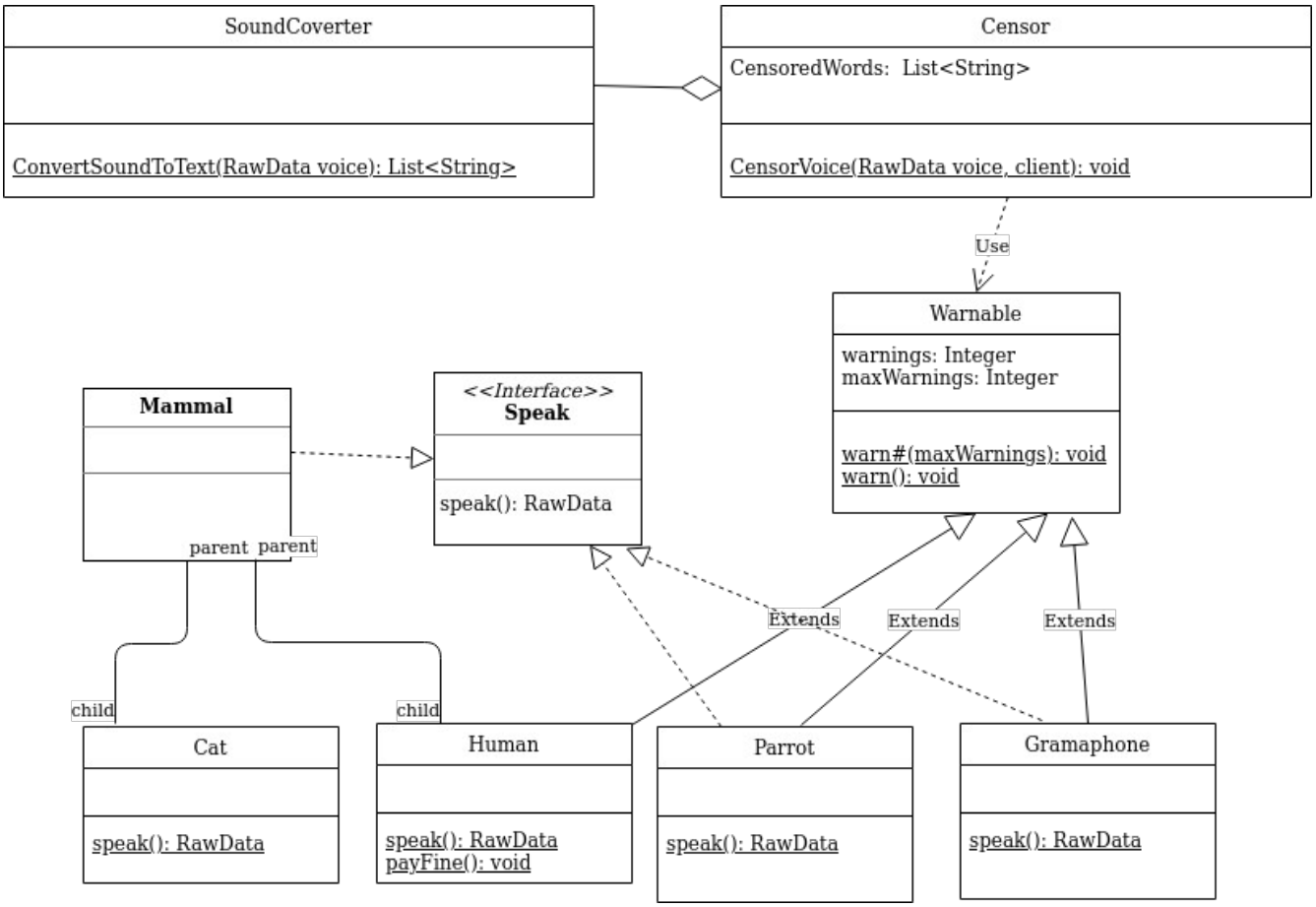
O: Extendable via interface without modification (more eaters/drinkers can be added, or more silverware, without modification of existing modules)

L: All instances implement the methods in their relevant interfaces, and knowledge of which instance is unnecessary

I: Eaters only exposed to Eater interface (with eat), Drinkers (with stir) and Silverware can access both stir and scoop

D: No higher level dependant on a lower level, or abstract dependant on details

Question 2



S: Each Module has a single responsibility :Convertor converts, censor censors, various speakers use speak interface

O: Extendable via interface without modification (more speakers/mammals etc can be added, without modification of existing modules)

L: All instances implement the methods in their relevant interfaces/or are extensions, and knowledge of which instance is unnecessary

I: No module is exposed to functions that it does not need to implement – for example only the human needs to pay a fine and therefore that method is unique to it

D: No higher level dependant on a lower level, or abstract dependant on details