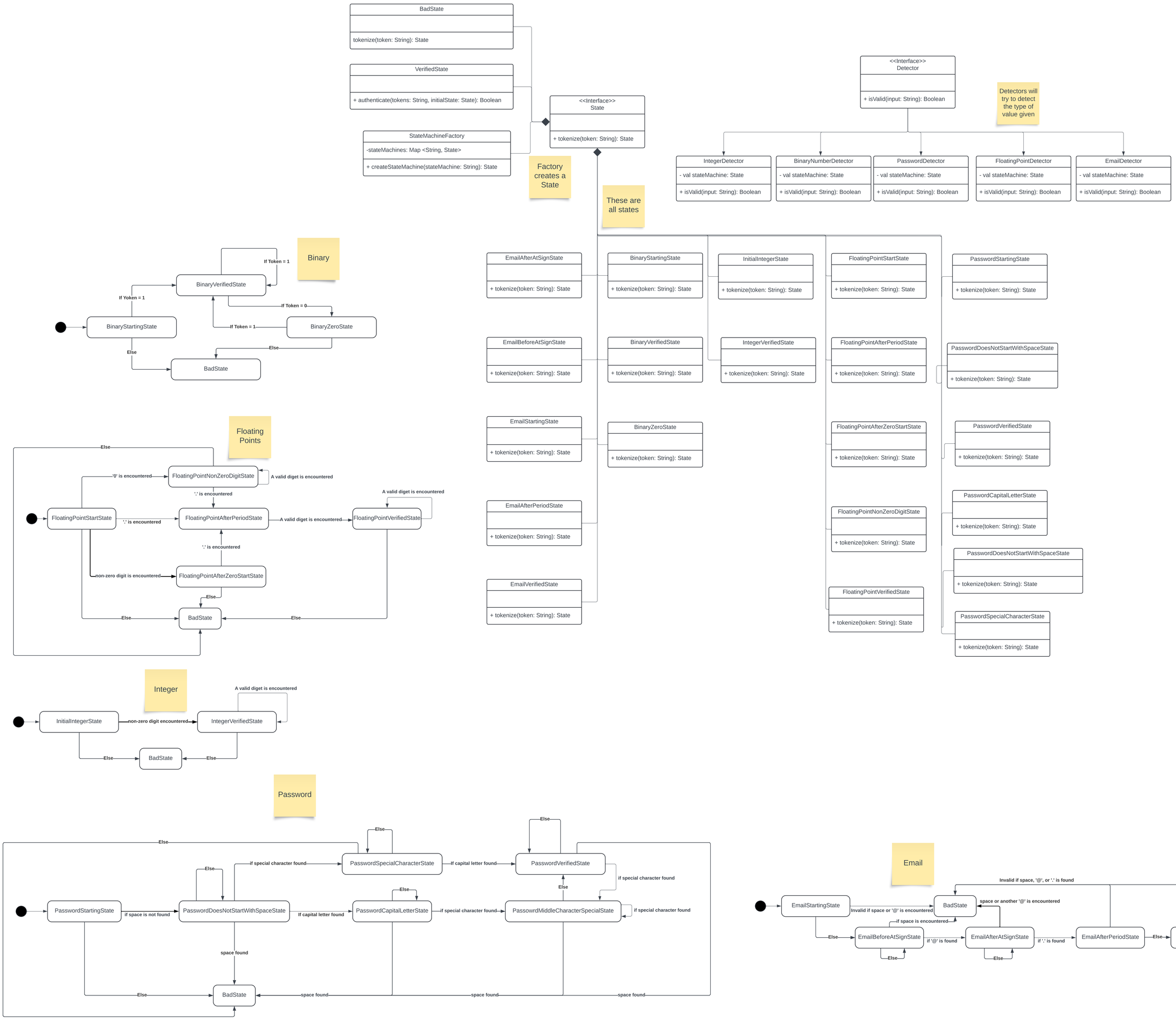
BadState
tokenize(token: String): State
VerifiedState
+ authenticate(tokens: String, initialState: State): Boolean
StateMachineFactory
-stateMachines: Map <String, State>
+ createStateMachine(stateMachine: String): State
Factory creates a State
<<Interface>>
State
+ tokenize(token: String): State
These are all states
<<Interface>>
Detector
+ isValid(input: String): Boolean
Detectors will try to detect the type of value given
IntegerDetector
- val stateMachine: State
+ isValid(input: String): Boolean
BinaryNumberDetector
- val stateMachine: State
+ isValid(input: String): Boolean
PasswordDetector
- val stateMachine: State
+ isValid(input: String): Boolean
FloatingPointDetector
- val stateMachine: State
+ isValid(input: String): Boolean
EmailDetector
- val stateMachine: State
+ isValid(input: String): Boolean
Binary
If Token = 1
BinaryVerifiedState
If Yoken = 1
If Token = 0
BinaryStartingState
If Token = 1
BinaryZeroState
Else
Else
BadState
EmailAfterAtSignState
+ tokenize(token: String): State
BinaryStartingState
+ tokenize(token: String): State
InitialIntegerState
+ tokenize(token: String): State
FloatingPointStartState
+ tokenize(token: String): State
PasswordStartingState
+ tokenize(token: String): State
EmailBeforeAtSignState
+ tokenize(token: String): State
BinaryVerifiedState
+ tokenize(token: String): State
IntegerVerifiedState
+ tokenize(token: String): State
FloatingPointAfterPeriodState
+ tokenize(token: String): State
PasswordDoesNotStartWithSpaceState
+ tokenize(token: String): State
EmailStartingState
+ tokenize(token: String): State
BinaryZeroState
+ tokenize(token: String): State
FloatingPointAfterZeroStartState
+ tokenize(token: String): State
PasswordVerifiedState
+ tokenize(token: String): State
EmailAfterPeriodState
+ tokenize(token: String): State
FloatingPointNonZeroDigitState
+ tokenize(token: String): State
PasswordCapitalLetterState
+ tokenize(token: String): State
PasswordDoesNotStartWithSpaceState
+ tokenize(token: String): State
EmailVerifiedState
+ tokenize(token: String): State
FloatingPointVerifiedState
+ tokenize(token: String): State
PasswordSpecialCharacterState
+ tokenize(token: String): State
Floating Points
Else
'0' is encountered
FloatingPointNonZeroDigitState
A valid diget is encountered
'.' is encountered
A valid diget is encountered
FloatingPointStartState
'.' is encountered
FloatingPointAfterPeriodState
A valid diget is encountered
FloatingPointVerifiedState
'.' is encountered
non-zero digit is encountered
FloatingPointAfterZeroStartState
Else
Else
BadState
Else
Integer
A valid diget is encountered
InitialIntegerState
non-zero digit encountered
IntegerVerifiedState
Else
BadState
Else
Password
Else
Else
Else
if special character found
PasswordSpecialCharacterState
If capital letter found
PasswordVerifiedState
if special character found
Else
Else
Else
PasswordStartingState
if space is not found
PasswordDoesNotStartWithSpaceState
If capital letter found
PasswordCapitalLetterState
if special character found
PassowrdMiddleCharacterSpecialState
if special character found
space found
Else
BadState
space found
space found
space found
Email
Invalid if space, '@', or '.' is found
EmailStartingState
Invalid if space or '@' is encountered
BadState
space or another '@' is encountered
if space is encountered
Else
EmailBeforeAtSignState
if '@' is found
EmailAfterAtSignState
if '.' is found
EmailAfterPeriodState
Else
EmailVerifiedState
Else
Else
Else