Class: **MicroslothActor**

Superclass: AdventureActor
Category: Adventure-Microsloth

I'm an AdventureActor that knows how to do a few Microsloth-specific things, such as eject players.

initialize-release

initialize

```
super initialize.
self pocket at: #slothToken put: MicroslothToken new
```

script actions

askSecurityGuardToEject: aPlayer

```
(self adventure componentAt: #securityGuard)
    eject: aPlayer for: self
```

askSecurityGuardToTerminate: aPlayer

```
(self adventure componentAt: #securityGuard)
    terminate: aPlayer for: self
```

dispenseSlothToken

```
self
    say: 'You''ll need this to get through the door.'
    or: 'Here is your Sloth for the door. Go right in.'
    or: 'You''ll need this for the door.'
    or: 'Use this to get through the door.'.
(Delay forSeconds: 2) wait.
(self pocket removeKey: #slothToken)
    container: self partner
```

script building

addPlayerReappearanceHandlerToWaitState: wState suspiciousState: sState hostileState: hState suspiciousResponseHolder: sHolder terminationState: tState

```
"wait"

    self addPlayerDisappearanceHandlerTo: wState
        successBlock: nil
        successState: sState.

    self addDefaultSpeechHandlersTo: wState.

"suspicious"

    self addPlayerAppearanceHandlerTo: sState
        successBlock: [:aScript :anEvent |
            self say: sHolder value]
        successState: hState.
```

```
"hostile"

	self addPlayerDisappearanceHandlerTo: hState
		successBlock: nil
		successState: sState.

	self addTimeoutHandlerTo: hState
		seconds: 10
		successBlock: [:aScript :anEvent |
			self say: 'You were warned!'
				or: 'You''re toast, pal!'
				or: 'I tried to warn you, but you wouldn''t listen!'
				or: 'You''re history...'.
			self askSecurityGuardToTerminate: self partner]
		successState: tState.

	self addDefaultSpeechHandlersTo: hState.
```

MetaClass: MicroslothActor class

Class: **MicroslothAdventure**

Superclass: Adventure
Category: Adventure-Microsloth

This is it, the Microsloth adventure. The key method here is #build.

initialize-release

build

```
| builder |
self id: #castleMicroSloth; name: 'Castle Microsloth'; description: 'A major time sink. '.
builder := AdventureBuilder on: self.
builder
    connect: #parkingLot
    and: #lobby
    via: #parkingLotToLobbyDoor
    and: #lobbyToParkingLotDoor.
builder
    connect: #lobby
    and: #hallway1
    via: #lobbyToHallway1Door
    and: #hallway1ToLobbyDoor.
builder
    connect: #hallway1
    and: #cappuccinoRoom
    via: #hallway1ToCappuccinoRoomDoor
    and: #cappuccinoRoomToHallway1Door.
builder
    connect: #hallway1
    and: #interviewRoom1
    via: #hallway1ToInterviewRoom1Door
    and: #interviewRoom1ToHallway1Door.
builder
    connect: #interviewRoom1
    and: #hallway2
    via: #interviewRoom1ToHallway2Door
    and: #hallway2ToInterviewRoom1Door.
builder
    connect: #hallway2
    and: #interviewRoom2
    via: #hallway2ToInterviewRoom2Door
    and: #interviewRoom2ToHallway2Door.
builder
    connect: #interviewRoom2
    and: #hallway3
    via: #interviewRoom2ToHallway3Door
    and: #hallway3ToInterviewRoom2Door.
builder
    connect: #hallway3
    and: #arcade
    via: #hallway3ToArcadeDoor
```

```
        and: #arcadeToHallway3Door.
    builder
        connect: #hallway3
        and: #interviewRoom3
        via: #hallway3ToInterviewRoom3Door
        and: #interviewRoom3ToHallway3Door.
    builder
        connect: #interviewRoom3
        and: #hallway4
        via: #interviewRoom3ToHallway4Door
        and: #hallway4ToInterviewRoom3Door.
    builder
        connect: #hallway4
        and: #executiveWashroom
        via: #hallway4ToExecutiveWashroomDoor
        and: #executiveWashroomToHallway4Door.
    builder
        connect: #hallway4
        and: #chairmansOffice
        via: #hallway4ToChairmansOfficeDoor
        and: #chairmansOfficeToHallway4Door.
    self startingPlace: (builder componentAt: #parkingLot)
```

lobby

briefcase

```
    | briefcase |
    briefcase := (AdventureLockingContainer
                id: #briefcase
                name: ’Briefcase’
                description: ’A black leather briefcase. ’)
                propertyAt: #lock put: (AdventureDeadboltLock shape: #nonExistantBriefcaseKey);
unlock.
    self visualAlgolBrochure container: briefcase.
    self dos621brochure container: briefcase.
    self executiveWashroomKey container: briefcase.
    MicroslothToken new container: briefcase.
    ^briefcase
```

dos621brochure

```
    ^AdventureBook
        id: #dos621brochure
        name: ’Brochure’
        description: ’A brochure for Microsloth DOS 6.2.1.7.3.2. A grinning yuppie is giving an
enthusiastic thumbs-up. The quotation below reads "With Microsloth True Data (C) 1:1 data compression,
I’’m using my WHOLE hard disk!" ’
```

executiveWashroomKey

```
    ^AdventureKey
        id: #executiveWashroomKey
        name: ’Washroom Key’
        description: ’A Yale ExSecure-2000 (C) Executive Washroom key. ’
```

lobby

```
    | lobby |
    lobby := AdventurePlace
```

```
                id: #lobby
                name: 'Lobby'
                description: 'The lobby at Microsloth Corporate HQ. '
                position: (AdventurePosition
                        x: 2
                        y: 1
                        z: 0).
    MicroslothReceptionist new container: lobby.
    MicroslothSecurityGuard new container: lobby.
    self wallClock container: lobby.
    self briefcase container: lobby.
    (AdventureHint string: 'If you read README.TXT, you know why you''re here!')
        container: lobby.
    ^lobby
```

lobbyToHallway1Door

```
    ^(AdventureTurnstile
        id: #lobbyToHallway1Door
        name: 'Door'
        description: 'A sign reads "1." ')
        propertyAt: #lock put: (AdventureTokenOperatedLock shape: #slothToken); yourself
```

lobbyToParkingLotDoor

```
    ^(AdventureLockingDoor
        id: #lobbyToParkingLotDoor
        name: 'Parking Lot Door'
        description: 'The front door leading to the parking lot. ')
        propertyAt: #lock put: AdventureEmergencyExitBar new; yourself
```

visualAlgolBrochure

```
    ^AdventureBook
        id: #visualAlgolBrochure
        name: 'Brochure'
        description: 'A brochure for Microsloth Visual Algol. The cover features a picture of a grad student
in front of a video monitor, looking at the camera with a quizzical expression on his face. He has a goatee,
and is wearing a black beret. A funny-looking cigarette dangles from his lips. A quote reads "Le Microsloth
Visual Algol? Oui!" '
```

wallClock

```
    ^(AdventureClock
        id: #lobbyWallClock
        name: 'Wall Clock'
        description: 'A wall clock with a graphical display. It is implemented using Wimpdoze For Clocks
3.1. ')
        isMoveable: false
```

parking lot

binaca

```
    ^AdventureThing
        id: #binaca
        name: 'Binaca'
        description: 'A small spray can of breath freshener. It''s the new "Pine Tree Ripple" flavor. Tasty! '
        value: 1
```

bmw990i

```
    | bmw990i |
    bmw990i := (AdventureLockingContainer
            id: #bmw990i
            name: 'BMW 990i'
            description: 'A black ' , (Date today year + 1) printString , ' BMW 990i. The Washington
state license plate reads "BILL." ')
            propertyAt: #lock put: (AdventureLatchingLock shape: #bmw990iKey);
            isMoveable: false;
            setImageFromBOSSFile: 'beamer.bos'.
    self bmw990iGloveBox container: bmw990i.
    ^bmw990i
```

bmw990iGloveBox

```
    | bmw990iGloveBox |
    bmw990iGloveBox := (AdventureClosingContainer
            id: #bmw990iGloveBox
            name: 'Glove Box'
            description: 'An ordinary BMW glove box. ')
            isMoveable: false.
    self queensrycheCD container: bmw990iGloveBox.
    self clearasil container: bmw990iGloveBox.
    self binaca container: bmw990iGloveBox.
    MicroslothToken new container: bmw990iGloveBox.
    ^bmw990iGloveBox
```

clearasil

```
    ^AdventureThing
        id: #clearasil
        name: 'Tube of Clearasil'
        description: 'A large tube of Clearasil squished into an impossible shape. Someone has been
squeezing from the middle. '
        value: 1
```

parkingLot

```
    | parkingLot |
    parkingLot := (AdventurePlace
            id: #parkingLot
            name: 'Parking Lot'
            description: 'The parking lot at Microsloth Corporate HQ. A sea of Hondas and Toyotas
seems to stretch for miles. Geese wander around aimlessly. '
            position: (AdventurePosition
                x: 2
                y: 0
                z: 0))
            setImageFromBOSSFile: 'parking.bos'.
    MicroslothParkingLotAttendant new container: parkingLot.
    self bmw990i container: parkingLot.
    (AdventureHint string: 'Lie.')
        container: parkingLot.
    ^parkingLot
```

parkingLotToLobbyDoor

```
    ^(AdventureTurnstile
        id: #parkingLotToLobbyDoor
        name: 'Front Door'
        description: 'The front door at Microsloth Corporation''s headquarters. It is flanked by a pair of
```

bamboo stakes. The heads of former customers perch there lifelessly. Their dead eyes are trying to tell you something... but what? ')
propertyAt: #lock put: (AdventureTokenOperatedLock shape: #slothToken); yourself

```
queensrycheCD
    ^(AdventureKey
        id: #queensrycheCD
        name: 'Queensryche CD'
        description: 'A compact disc. It''s a copy of the new Queensryche release "Wish We Were Floyd."
'
        value: 13)
        setImageFromBOSSFile: 'cd.bos'
```

hallway 1

```
hallway1
    | hallway1 |
    hallway1 := AdventureHall
            id: #hallway1
            name: 'Hallway #1'
            description: 'Like it says, this is Hallway #1. '
            position: (AdventurePosition
                    x: 2
                    y: 2
                    z: 0).
    ^hallway1
```

```
hallway1ToCappuccinoRoomDoor
    ^AdventureSwingingDoor
        id: #hallway1ToCappuccinoRoomDoor
        name: 'Swinging Door'
        description: 'A sign reads "Employees Only." The smell of high-powered coffee wafts out. Maybe
you''d like a little something to sharpen you up for the interview? '
```

```
hallway1ToInterviewRoom1Door
    ^AdventureDoor
        id: #hallway1ToInterviewRoom1Door
        name: 'Door'
        description: 'A sign reads "Interview 1." '
```

```
hallway1ToLobbyDoor
    ^(AdventureLockingDoor
        id: #hallway1ToLobbyDoor
        name: 'Door'
        description: 'A sign reads "Lobby." ')
        propertyAt: #lock put: AdventureEmergencyExitBar new; yourself
```

hallway 2

```
hallway2
    | hallway2 |
    hallway2 := AdventureHall
            id: #hallway2
            name: 'Hallway #2'
            description: 'Like it says, this is Hallway #2. '
```

```
			position: (AdventurePosition
				x: 2
				y: 4
				z: 0).
	^hallway2
```

hallway2ToInterviewRoom1Door

```
	^(AdventureLockingDoor
		id: #hallway2ToInterviewRoom1Door
		name: 'Door'
		description: 'A sign reads "Exit." ')
		propertyAt: #lock put: AdventureEmergencyExitBar new; yourself
```

hallway2ToInterviewRoom2Door

```
	^AdventureDoor
		id: #hallway2ToInterviewRoom2Door
		name: 'Door'
		description: 'A sign reads "Interview 2." '
```

hallway 3

hallway3

```
	| hallway3 |
	hallway3 := AdventureHall
			id: #hallway3
			name: 'Hallway #3'
			description: 'Like it says, this is Hallway #3. '
			position: (AdventurePosition
				x: 2
				y: 6
				z: 0).
	^hallway3
```

hallway3ToArcadeDoor

```
	^AdventureDoor
		id: #hallway3ToArcadeDoor
		name: 'Door'
		description: 'A sign reads "Recreation." '
```

hallway3ToInterviewRoom2Door

```
	^(AdventureLockingDoor
		id: #hallway3ToInterviewRoom2Door
		name: 'Door'
		description: 'A sign reads "Exit." ')
		propertyAt: #lock put: AdventureEmergencyExitBar new; yourself
```

hallway3ToInterviewRoom3Door

```
	^AdventureDoor
		id: #hallway3ToInterviewRoom3Door
		name: 'Door'
		description: 'A sign reads "Interview 3." '
```

hallway 4

hallway4

```
| hallway4 |
hallway4 := AdventureHall
        id: #hallway4
        name: 'Hallway #4'
        description: 'Like it says, this is Hallway #4. '
        position: (AdventurePosition
            x: 2
            y: 8
            z: 0).
^hallway4
```

hallway4ToChairmansOfficeDoor

```
^AdventureDoor
    id: #hallway4ToChairmansOffice
    name: 'Door'
    description: 'A massive door, machined out of a single block of titanium. A sign reads "His Excellency, Bill." '
```

hallway4ToExecutiveWashroomDoor

```
^(AdventureTurnstile
    id: #hallway4ToExecutiveWashroomDoor
    name: 'Washroom Door'
    description: 'A sign reads "Executives Only." You wonder if they have those neat little bars of soap. ')
    propertyAt: #lock put: (AdventureLatchingLock shape: #executiveWashroomKey); yourself
```

hallway4ToInterviewRoom3Door

```
^(AdventureLockingDoor
    id: #hallway4ToInterviewRoom3Door
    name: 'Door'
    description: 'A sign reads "Exit." ')
    propertyAt: #lock put: AdventureEmergencyExitBar new; yourself
```

interview room 1

interviewRoom1

```
| interviewRoom1 |
interviewRoom1 := AdventurePlace
        id: #interviewRoom1
        name: 'Interview Room #1'
        description: 'Like it says, this is Interview Room #1. '
        position: (AdventurePosition
            x: 2
            y: 3
            z: 0).
MicroslothInterviewer1 new container: interviewRoom1.
(AdventureHint string: 'Have you read your interviewer''s bio?')
    container: interviewRoom1.
^interviewRoom1
```

interviewRoom1ToHallway1Door

```
^AdventureDoor
    id: #interviewRoom1ToHallway1Door
    name: 'Door'
    description: 'A sign reads "1". '
```

interviewRoom1ToHallway2Door

```
^(AdventureTurnstile
    id: #interviewRoom1ToHallway2Door
    name: 'Door'
    description: 'A sign reads "2". ')
    propertyAt: #lock put: (AdventureTokenOperatedLock shape: #slothToken); yourself
```

interview room 2

interviewRoom2

```
| interviewRoom2 |
interviewRoom2 := AdventurePlace
            id: #interviewRoom2
            name: 'Interview Room #2'
            description: 'Like it says, this is Interview Room #2. '
            position: (AdventurePosition
                    x: 2
                    y: 5
                    z: 0).
MicroslothInterviewer2 new container: interviewRoom2.
(AdventureHint string: 'Does the term "MicroSerf" mean anything to you?')
    container: interviewRoom2.
^interviewRoom2
```

interviewRoom2ToHallway2Door

```
^AdventureDoor
    id: #interviewRoom2ToHallway2Door
    name: 'Door'
    description: 'A sign reads "2". '
```

interviewRoom2ToHallway3Door

```
^(AdventureTurnstile
    id: #interviewRoom2ToHallway3Door
    name: 'Door'
    description: 'A sign reads "3". ')
    propertyAt: #lock put: (AdventureTokenOperatedLock shape: #slothToken); yourself
```

interview room 3

interviewRoom3

```
| interviewRoom3 |
interviewRoom3 := AdventurePlace
            id: #interviewRoom3
            name: 'Interview Room #3'
            description: 'Like it says, this is Interview Room #3. '
            position: (AdventurePosition
                    x: 2
                    y: 7
                    z: 0).
MicroslothInterviewer3 new container: interviewRoom3.
(AdventureHint string: 'This interviewer''s opinions do not necessarily represent those of Microsloth
Corporation...')
    container: interviewRoom3.
^interviewRoom3
```

interviewRoom3ToHallway3Door

```
^AdventureDoor
    id: #interviewRoom3ToHallway3Door
    name: ’Door’
    description: ’A sign reads "3". ’
```

interviewRoom3ToHallway4Door

```
^(AdventureTurnstile
    id: #interviewRoom3ToHallway4Door
    name: ’Door’
    description: ’A sign reads "4". ’)
    propertyAt: #lock put: (AdventureTokenOperatedLock shape: #slothToken); yourself
```

cappuccino room

annualReport

```
^AdventureBook
    id: #annualReport
    name: ’Annual Report’
    description: ’It’’s the Microsloth Annual Report. It features a screen shot of Wimpdoze 4.0:
"Beirut." A caption reads "artist’’s conception." ’
```

cappuccinoRoom

```
| cappuccinoRoom |
cappuccinoRoom := AdventurePlace
        id: #cappuccinoRoom
        name: ’Cappuccino Room’
        description: ’An ordinary room, about 12 feet square, with a table and chairs in the
middle. ’
        position: (AdventurePosition
            x: 1
            y: 2
            z: 0).
self annualReport container: cappuccinoRoom.
self menOfSlothCalendar container: cappuccinoRoom.
self womenOfSlothCalendar container: cappuccinoRoom.
self microslothNewsletter container: cappuccinoRoom.
self sayingsOfChairmanBill container: cappuccinoRoom.
self seattleTimes container: cappuccinoRoom.
((MicroslothCappuccino destinationComponentId: #hallway1)
    isEmpty: true)
    container: cappuccinoRoom.
(MicroslothCappuccinoMachine destinationComponentId: #hallway1)
    container: cappuccinoRoom.
(AdventureHint string: ’Thirsty?’)
    container: cappuccinoRoom.
^cappuccinoRoom
```

cappuccinoRoomToHallway1Door

```
^AdventureInoperableDoor
    id: #cappuccinoRoomToHallway1Door
    name: ’Door’
    description: ’It appears to have no opening mechanism. A sign reads "No Exit." ’
```

menOfSlothCalendar

```
^(AdventureBook
```

```
        id: #menOfSlothCalendar
        name: ’Calendar: Men of Sloth’
        description: ’Security Guard "Spike" Swindell graces the cover. He is flexing. ’)
        isMoveable: false
```

microslothNewsletter

```
    ^AdventureBook
        id: #microslothNewsletter
        name: ’Newsletter’
        description: ’The latest copy of the company newsletter. Big headlines: "Marmots Lose," "New
Sloth Depicts Mature Lord Bill," and "Kids Visit Bill’’s Birthplace." ’
```

sayingsOfChairmanBill

```
    ^AdventureBook
        id: #sayingsOfChairmanBill
        name: ’Book’
        description: ’A copy of Bill’’s bestseller "The Sayings of Chairman Bill," by Microsloth Press. It’’s
open to Saying 324: "There will always be a man who can make something a little cheaper, a little
shabbier, for the sake of a buck. I am that man..." ’
```

seattleTimes

```
    ^AdventureBook
        id: #seattleTimes
        name: ’Newspaper’
        description: ’A copy of the Seattle Times. It’’s open to the "Help Wanted" section, at the listings
for Food Service. ’
```

womenOfSlothCalendar

```
    ^(AdventureBook
        id: #womenOfSlothCalendar
        name: ’Calendar: Women of Sloth’
        description: ’This month’’s entry features smiling receptionist Tiffany in a tight Microsloth Marmots
T-shirt. ’)
        isMoveable: false
```

arcade

arcade

```
    | arcade |
    arcade := AdventurePlace
                id: #arcade
                name: ’Arcade’
                description: ’A room full of video games. ’
                position: (AdventurePosition
                        x: 1
                        y: 6
                        z: 0).
    (MicroslothVideoGame withTokens: 1)
        isWorking: false;
        container: arcade.
    (MicroslothVideoGame new) isWorking: false; container: arcade.
    MicroslothVideoGame new container: arcade.
    (MicroslothTelevision withDialog: self arcadeTelevisionDialog) container: arcade.
    (AdventureHint string: ’Persistence is a virtue.’)
        container: arcade.
    ^arcade
```

arcadeTelevisionDialog

```
    ^#(
        'Welcome to the Microsloth Sports Network.'
        'Coming up, today''s game between the Microsloth Marmots, and the Lotus Lemmings.'
        'But first, the Microsloth National Anthem, sung by Mr. David Hasslehoff, and his Babe Watch (C)
Friends.'
        '(ahem)'
        'God save Lord Bill;'
        'Send him vic-to-ri-ous,'
        'Hap-py and glo-ri-ous,'
        'Long to reign o-ver us;'
        'God save Lord Bill.'
        'God save our gra-cious Bill!'
        'Long live our no-ble Bill!'
        '(uhumm)'
        'O Lord our Bill arise;'
        'Scatter your en-em-ies,'
        'And make them fall,'
        'Confound their po-li-tics,'
        'Frustrate their knavish tricks,'
        'On thee our hopes we fix,'
        'Bill save us all.'
        'God save our gra-cious Bill!'
        'Long live our no-ble Bill!'
        "'(uhemm)'
        'Thy choi-cest gifts in store,'
        'On him be pleased to pour,'
        'Long may he reign,'
        'May he de-fend our laws,'
        'And ever give us cause,'
        'To sing with heart and voice,'
        'God save Lord Bill.'"
    )
```

arcadeToHallway3Door

```
    ^AdventureDoor
        id: #arcadeToHallway3Door
        name: 'Door'
        description: 'A sign reads "Exit." '
```

executive washroom

clearasilDispenser

```
    ^(AdventureThing
        id: nil
        name: 'Clearasil Dispenser'
        description: 'An institution-size Clearasil dispenser. The clear window on the side shows half full.
')
        isMoveable: false
```

executiveSoap

```
    ^AdventureThing
        id: nil
        name: 'Bar of Soap'
```

description: ’A bar of "Executive Bouquet - Be an Executive... or Just Smell Like One!" ’

executiveWashroom

```
| executiveWashroom |
executiveWashroom := (AdventurePlace
        id: #executiveWashroom
        name: ’Executive Washroom’
        description: ’This is where the executives catch up on their reading... ’
        position: (AdventurePosition
            x: 1
            y: 8
            z: 0))
        setImageFromBOSSFile: ’washroom.bos’.
self washroomStall container: executiveWashroom.
self washroomSink container: executiveWashroom.
(AdventureHint string: ’When ya gotta go, ya gotta go...’)
    container: executiveWashroom.
^executiveWashroom
```

executiveWashroomToHallway4Door

```
^(AdventureLockingDoor
    id: #executiveWashroomToHallway4Door
    name: ’Door’
    description: ’A sign reads "Exit." ’)
    propertyAt: #lock put: AdventureEmergencyExitBar new; yourself
```

minutesOfMarketingMeeting

```
^AdventureBook
    id: nil
    name: ’Minutes of Marketing Meeting’
    description: (’It’’s a document entitled:\"Executive Marketing Council\minutes of ’ , Date today printString , ’"\It discusses selection of a code name for the newest flavor of Wimpdoze: "Wimpdoze for Workcamps." Apparently, Mainland China is anxious to employ slave labor in the production of software. They have contracted Microsloth to provide the ideal development environment for the imprisoned intellectual. In part, it reads:\Bill G: So what have we got?\Mike L: "Kampala," "Capetown," and "Gulag."\Bill: I don’’t like it. Nothing captures the essence of the Wimpdoze for Workcamps EXPERIENCE, you know?\Lisa B: What about "Los Angeles?"\Eric M: You’’re forgetting--that’’s being used for Windows for Cellphones...\Lisa B: What about the Balkans? Albania, maybe?\Mike L: Maybe something from World War II. You know, something that conjures up 40’’s hair do’’s and Benny Goodman.\Bill G: No! This is a state of the art product. Something current. Hip. Now... Hmmm...\Bill G: I’’ve got it: Wimpdoze for Workcamps, code-named: "Sarajevo!"\(group): Hail! Hail! Hail!...’) withCRs
```

washroomSink

```
| washroomSink |
washroomSink := (AdventureSimpleContainer
        id: nil
        name: ’Washroom Sink’
        description: ’A stainless-steel bathroom sink. A digital faucet runs Wimpdoze for Bathrooms 3.1. It appears to be "down." The faucet taps out the morse-code for "GPF ERROR"; hot water for "dash," cold water for "dot." ’)
        isMoveable: false.
self executiveSoap container: washroomSink.
self clearasilDispenser container: washroomSink.
^washroomSink
```

washroomStall

```
    | washroomStall |
    washroomStall := (AdventureLockingContainer
            id: nil
            name: 'Washroom Stall'
            description: 'A white enamel bathroom stall. ')
            propertyAt: #lock put: (AdventureTokenOperatedLock shape: #slothToken);
            isMoveable: false.
    self minutesOfMarketingMeeting container: washroomStall.
    ^washroomStall
```

chairmans office

chairmansOffice

```
    | chairmansOffice |
    chairmansOffice := (AdventurePlace
            id: #chairmansOffice
            name: 'Chairman''s Office'
            description: 'An ordinary office (with a door!) Pretty austere stuff for a man with "infinite
money."'
            position: (AdventurePosition
                x: 2
                y: 9
                z: 0))
            setImageFromBOSSFile: 'office.bos'.
    MicroslothChairmanBill new container: chairmansOffice.
    (AdventureHint string: 'Have you kept up with your reading?')
        container: chairmansOffice.
    ^chairmansOffice
```

chairmansOfficeToHallway4Door

```
    ^AdventureInoperableDoor
        id: #chairmansOfficeToHallway4Door
        name: 'Door'
        description: 'A smooth, featureless titanium door on massive hinges. It appears to have no
opening mechanism. A sign reads "Abandon Hope" '
```

interface opening

postOpenWith: anAdventureControlPanel

```
    super postOpenWith: anAdventureControlPanel.
    "(self componentAt: #receptionist) enableSingleStepping."
    "(self componentAt: #parkingLotAttendant) enableScriptTrace."
    "AdventureDetonator openOn: self"
```

MetaClass: MicroslothAdventure class

product

productAboutText

```
    ""
    "self flushAllProductCaches"
```

```
    ^Text fromString:
'Authors
Design and Programming by Adam Shackleford (approach@aol.com). Graphics by Susan Atherly
(suediane@aol.com).

The Adventure
Castle Microsloth is a graphical, realtime adventure written in ParcPlace VisualWorks. It''s implemented
with a toolbox of 100+ adventure-building classes. Also incorporated are color icons, custom-drawn for
the game by Susan Atherly (suediane@aol.com).

The Objective
Your character is applying for a job at a well-known software firm in the Pacific Northwest. Your objective:
survive the interview process and score Your Own Office (with a door!).

Special Thanks
Special thanks to the following supporters: Laszlo Takacs (play testing,) Susan Atherly (graphics, game
design and play testing,) Rik Smoody (play testing and UI suggestions,) David Reeder (play testing.)

Consulting Services
To inquire about my Smalltalk consulting services, please send email to approach@aol.com

Copyright
This software Copyright (C) 1994 by Adam E. Shackleford. All Rights Reserved. Graphics courtesy of
Susan D. Atherly. Copyright (C) 1994 by Susan D. Atherly. All Rights Reserved.

License
You are licensed to use this software under the terms of the GNU General Public License Version 2, June
1991. The text of that license is included in the distribution.



VisualWorks is a trademark of ParcPlace Systems. '
```

productCopyright

```
    ""
    "self flushAllProductCaches"

    ^'Copyright (C) 1994 by Adam E. Shackleford. All Rights Reserved. Graphics courtesy of Susan D.
Atherly. Copyright (C) 1994 by Susan D. Atherly. All Rights Reserved.'
```

productCredits

```
    ""
    "self flushAllProductCaches"

    ^Array with: (AdventureCredit
            role: 'Design and Programming'
            name: 'Adam Shackleford'
            additionalInformation: '(approach@aol.com)')
        with: (AdventureCredit
                role: 'Graphics'
                name: 'Susan Atherly'
                additionalInformation: '(suediane@aol.com)')
```

productLogo

```
    ""
```

```
"self flushAllProductCaches"

^CachedImage on: (Adventure readImageFromBOSSFile: 'guard.bos')
```

productName

```
""
"self flushAllProductCaches"

^'Castle Microsloth'
```

productVersion

```
""
"self flushAllProductCaches"

^'3.1a'
```

view creation

defaultPlayer

```
^AdventurePlayer
    id: #player
    name: 'Lee Q. Player'
    description: 'Hi, I''m Lee, your character in this Adventure. I''m desperate for a chance to work side by side with industry pioneer Chairman Bill! '
```

defaultUserInterface

```
^MicroslothAdventureControlPanel
```

testing

isStartable

```
^true
```

saving

binaryRepresentationVersion

```
^'Castle Microsloth 3.1a'
```

Class: **MicroslothAdventureControlPanel**

Superclass: AdventureControlPanel
Category: Adventure-Microsloth

This is a version of the adventure UI with a little customization.

player outcomes

displayPlayerLossDialog
MicroslothLossDialog open

displayPlayerWinDialog
MicroslothWinDialog open

MetaClass: MicroslothAdventureControlPanel class

Class: **MicroslothCappuccino**

Superclass: AdventureThing
Category: Adventure-Microsloth
Instance variables: destinationComponentId isEmpty
Class variables: CappuccinoDisorientationResponses
RevoltingDregsResponses

I'm a tasty cup of Microsloth's own cappuccino. If you drink me, I transport you to the adventure component corresponding to my #destinationComponentId.

Instance Variables:
destinationComponentId <Symbol> The id of the container where you'll go if you drink me. At run time, this is used to ask the adventure for the actual container.
isEmpty<Boolean> True if I have been consumed. This renders me harmless.

Class Variables:
CappuccinoDisorientationResponses <AdventureRandomValueHolder> of String's to say after consuming a cup of me.
RevoltingDregsResponses <AdventureRandomValueHolder> of String's to say after licking the dregs of me out of a cup.

menu items

addMenuItemsFor: aPlayer to: aCollection
```
aCollection add: (self isEmpty ifFalse: [AdventureMenuItem label: 'drink'
            block:
                [aPlayer container: (aPlayer adventure componentAt: self destinationComponentId).
                aPlayer say: self cappuccinoDisorientationResponses value.
                self isEmpty: true]]
        ifTrue: [AdventureMenuItem label: 'lick the dregs' block: [aPlayer say: self
revoltingDregsResponses value]]).
super addMenuItemsFor: aPlayer to: aCollection
```

initialize-release

destinationComponentId: aSymbol
```
destinationComponentId := aSymbol
```

initialize
```
super initialize.
self name: 'Cup of Cappuccino'.
self description: 'A cup of high-test cappuccino. If you put your ear to the cup, you can hear a faint
high-pitched whine. Lettering on the side says "Drink Me." '
```

accessing

cappuccinoDisorientationResponses
```
""
"CappuccinoDisorientationResponses := nil"
```

```
    ^CappuccinoDisorientationResponses isNil
        ifTrue: [CappuccinoDisorientationResponses := AdventureRandomValueHolder
                withAll: #(
                    'Whoa! Where am I?'
                    'I feel kinda funny. Where am I?'
                    'What the...! Where am I?'
                    'Wow. Awesome buzz.'
                    'Hokey smokes, Bullwinkle, that''s good cappuccino!'
                    'Wow! Tres wicked brew. Hey, where the heck am I?'
                    'Did anyone get the name of that truck?'
                    'I think I just saw God... He was wearing glasses and jumping over a chair.'
                )]
        ifFalse: [CappuccinoDisorientationResponses]
```

description

```
    ^super description , (self isEmpty
            ifTrue: ['It is empty. ']
            ifFalse: ['It is full. '])
```

destinationComponentId

```
    ^destinationComponentId
```

isEmpty

```
    ^isEmpty isNil
        ifTrue: [isEmpty := false]
        ifFalse: [isEmpty]
```

isEmpty: aBoolean

```
    isEmpty := aBoolean
```

revoltingDregsResponses

```
    ""
    "RevoltingDregsResponses := nil"

    ^RevoltingDregsResponses isNil
        ifTrue: [RevoltingDregsResponses := AdventureRandomValueHolder
                withAll: #(
                    'Yeccch!'
                    'Eeeeyoo!'
                    'Yuck!'
                    'Ick!'
                    'Ickk. Did someone use this for an ashtray, or what?'
                    'Urrgh. I think I''m gonna hurl.'
                    'Ooooo. That was seriously grim.'
                )]
        ifFalse: [RevoltingDregsResponses]
```

MetaClass: MicroslothCappuccino class

instance creation

destinationComponentId: aSymbol

```
^self new destinationComponentId: aSymbol
```

Class: **MicroslothCappuccinoMachine**

Superclass: AdventureSimpleContainer
Category: Adventure-Microsloth

I dispense magical cappuccino that sends the drinker to the destination specified by the destinationComponentId that I was initialized with.

initialize-release

destinationComponentId: aSymbol

```
    self propertyAt: #mechanism put: (MicroslothCappuccinoMachineMechanism
destinationComponentId: aSymbol)
```

initialize

```
    self name: 'Cappuccino Machine'.
    self description: 'A stainless steel automated cappuccino vending machine. '.
    self isMoveable: false
```

MetaClass: MicroslothCappuccinoMachine class

instance creation

destinationComponentId: aSymbol

```
    ^self new destinationComponentId: aSymbol
```

Class: **MicroslothCappuccinoMachineMechanism**

Superclass: AdventureTokenOperatedMechanism
Category: Adventure-Microsloth

I accept a token, then produce a cup of magical cappuccino. Drink the cappuccino, and you (the player) will be transported to the container that corresponds to the destinationComponentId that I was initialized with.

I assume that my #owner is a kind of AdventureSimpleContainer, since I dispense a cup of cappuccino into my owner when actuated.

initialize-release

destinationComponentId: aSymbol

```
self actuateBlock: [:aPlayer | ((MicroslothCappuccino destinationComponentId: aSymbol)
        adventure: self owner adventure)
        container: self owner]
```

initialize

```
super initialize.
self shape: #slothToken
```

MetaClass: MicroslothCappuccinoMachineMechanism class

instance creation

destinationComponentId: aSymbol

```
^self new destinationComponentId: aSymbol
```

Class: **MicroslothChairmanBill**

Superclass: MicroslothInterviewer
Category: Adventure-Microsloth

I am the Chairman of Microsloth Corporation. I have infinite money.

initialize-release

initialize

```
	super initialize.
	self
		id: #chairmanBill
		name: 'Chairman Bill'
		description: 'Microsloth Corp''s President and CEO. He is dressed in a black cape, jumpsuit, and jack boots. A shiny black helmet completely covers his head. A black Dust Buster dangles carelessly from one hip. '
		biography: 'Born to loving parents of modest means, Bill was just another Harvard drop-out, sharecropping to make ends meet, until that fateful day in 1981. As Bill strained against his plow at the end of another day of fruitless toil, a mighty blue meteorite split the sky with a thunderous clap. Bill cautiously approached the smoking extra-terrestrial visitor. He was shocked to find six-billion U.S. dollars and a really ugly pair of eyeglasses. The rest, as they say, is history... '
```

script building

buildScript

```
	| myScript wrongAnswerScoldings |
	myScript := (AdventureScript named: #myScriptName actor: self)
		add: (AdventureScriptState named: #waitingForPlayer actor: self);
		add: (AdventureScriptState named: #youMadeIt actor: self);
		add: (AdventureScriptState named: #suggestACodeName actor: self);
		yourself.

	wrongAnswerScoldings := AdventureRandomValueHolder withAll: #(
		'That sucks.'
		'Are you brain dead?'
		'Try again, Einstein.'
		'Boring.'
		'Is that the best you can do? Try again!'
		'Nein!'
		'Pitiful.'
		'That''s really bad.'
		'That''s stupid.'
		'That''s pretty idiotic.'
		'Weak. Very weak.'
		'If you only had a brain...'
		'Nice try. Now try again.'
		'Really terrible.'
		'Hey. That really stinks!'
		'Exceptionally bad.'
```

```
	).

"waitingForPlayer"

	self addPlayerAppearanceHandlerTo: (myScript atState: #waitingForPlayer)
		successBlock: nil
		successState: (myScript atState: #youMadeIt).

"youMadeIt"

	self addQuestionHandlerTo: (myScript atState: #youMadeIt)
		questions: #(
			'Well, you''ve made it this far. What do you think so far?'
			'Well Lee, I didn''t think you''d make it. How do you feel?'
			'You made it. I guess you lucked out. What do you think of Microsloth so far?'
			'You made it. Incredible. So what d''ya think?'
		)
		answers: (
			Array with: (
				AdventureAnswer
					cue: (
						AdventureSpeechCue
							machineEvent: (MachineEvent named: #playerReplied)
							actor: self
							matcher: ('*')
					)
					actionBlock: [:aScript :anEvent |
						self say: 'Typical answer.'
							or: 'Good answer. Not.'
							or: 'Yeah, that''s really great.'
							or: 'Uh huh. Same answer the last guy gave.'.
						self say: 'Since you made it this far, I won''t waste your time with any more
questions about memory leaks or traversing priority queues.'.
						self say: 'However, you might actually be of some use.'.
						self say: 'We''re developing a new version of Wimpdoze: "Wimpdoze for
Workcamps 3.1" for an undisclosed Asian customer.'.
						self say: 'It''s optimized for a networked camp of imprisoned intellectuals.'.
						self say: 'They''re being used as slave labor to develop software.'.
						self say: 'Heck, if they work out, we''ll be farming-out lots of stuff.'.
						self say: 'I mean, these guys''ll work for a bowl of rice a day!']
					nextState: (myScript atState: #suggestACodeName)
			)
		)
		disappearanceBlock: nil
		disappearanceState: (myScript atState: #waitingForPlayer)
		failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
		failureState: (myScript atState: #waitingForPlayer).

"suggestACodeName"

	self addQuestionHandlerTo: (myScript atState: #suggestACodeName)
		questions: #(
			'Anyway, we need a cool code-name to leak to the press. Named after a city. I have
something in mind. Why don''t you make a few suggestions.'
		)
		answers: (
```

```
        Array with: (
            AdventureAnswer
                cue: (
                    AdventureSpeechCue
                        machineEvent: (MachineEvent named: #rightAnswer)
                        actor: self
                        matcher: (
                            AdventureWordMatcher withAll:
                                #('Sara*')
                        )
                )
                actionBlock: [:aScript :anEvent | self partner win]
                nextState: (myScript atState: #waitingForPlayer)
        )
    )
    timeoutResponseHolder: self timeoutResponses
    nonsenseResponseHolder: wrongAnswerScoldings
    obscenityResponseHolder: self obscenityResponses
    chances: self defaultChances * 2
    timeout: self defaultTimeout
    disappearanceBlock: nil
    disappearanceState: (myScript atState: #waitingForPlayer)
    failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
    failureState: (myScript atState: #waitingForPlayer).

^myScript
```

MetaClass: MicroslothChairmanBill class

resources

buildImage

```
^CachedImage on: (Adventure readImageFromBOSSFile: 'billg.bos')
```

Class: **MicroslothDOSCommandLine**

Superclass: ApplicationModel
Category: Adventure-Microsloth
Instance variables: semaphore displayTime

I am a convincing replica of a window containing a DOS command line. I can be opened for a given number of milliseconds, after which time, I close myself.

Instance Variables:
displayTime<Integer> Number of milliseconds that I display.
semaphore <Semaphore> I use this to keep from trying to close the window before it has been opened, regardless of how short displayTime is.

interface opening

openTimed

```
"Without the second #fork, we don't update the window."

self semaphore: Semaphore new.

[[self open] fork.
self semaphore wait.
(Delay forMilliseconds: self displayTime) wait.
self closeRequest] fork
```

postOpenWith: aBuilder

```
self semaphore signal
```

accessing

displayTime

```
^displayTime isNil
    ifTrue: [displayTime := 4000]
    ifFalse: [displayTime]
```

displayTime: anInteger

```
displayTime := anInteger
```

semaphore

```
^semaphore
```

semaphore: aSemaphore

```
semaphore := aSemaphore
```

view creation

openForMilliseconds: anIntegerOrNil

```
self forMilliseconds: anIntegerOrNil.
self openTimed
```

initialize-release

forMilliseconds: anIntegerOrNil

```
self displayTime: anIntegerOrNil
```

MetaClass: MicroslothDOSCommandLine class

interface specs

windowSpec

```
"UIPainter new openOnClass: self andSelector: #windowSpec"

^#(#FullSpec #window: #(#WindowSpec #label: 'DOS' #min: #(#Point 565 335 ) #max: #(#Point 565
335 ) #bounds: #(#Rectangle 318 355 883 690 ) ) #component: #(#SpecCollection #collection:
#(#(#LabelSpec #layout: #(#Point -3 -3 ) #label: 'dosImage' #hasCharacterOrientedLabel: false ) ) ) )
```

resources

dosImage

```
^CachedImage on: (Adventure readImageFromBOSSFile: 'dos.bos')
```

view creation

openForMilliseconds: anIntegerOrNil

```
^self new openForMilliseconds: anIntegerOrNil
```

Class: **MicroslothInformationDialog**

Superclass: SimpleDialog
Category: Adventure-Microsloth
Instance variables: text buttonLabel

I'm a general-purpose information dialog. You specify a buttonLabel: and a text: string. I display the icon of a randomly-selected character, the text string, and a labeled button.

Instance Variables:
text <String> to display in my text box.
buttonLabel <String> to label my button with.

Class Instance Variables:
logoHolder <AdventureRandomValueHolder> of CachedImage's to display as my logo image.

interface opening

postBuildWith: aBuilder

```
(aBuilder componentAt: #button)
    labelString: self buttonLabel.
(aBuilder componentAt: #logo)
    label: self class logoHolder value
```

accessing

buttonLabel

```
^buttonLabel
```

buttonLabel: aString

```
buttonLabel := aString
```

text

```
^text
```

text: aString

```
text := aString
```

initialize-release

buttonLabel: bString text: tString

```
self buttonLabel: bString; text: tString
```

MetaClass: MicroslothInformationDialog class

Instance variables: logoHolder

interface specs

windowSpec

```
    "UIPainter new openOnClass: self andSelector: #windowSpec"

    ^#(#FullSpec #window: #(#WindowSpec #label: 'Information Dialog' #min: #(#Point 20 20 ) #bounds: 
#(#Rectangle 349 349 647 526 ) ) #component: #(#SpecCollection #collection: #(#(#ActionButtonSpec 
#layout: #(#Rectangle 8 120 288 168 ) #name: #button #model: #cancel #label: 'button' #defaultable: true 
) #(#LabelSpec #layout: #(#Point 8 8 ) #name: #logo #label: 'logo' ) #(#TextEditorSpec #layout: 
#(#Rectangle 80 8 288 104 ) #flags: 4 #model: #text #isReadOnly: true ) ) ) )
```

view creation

buttonLabel: bString text: tString

```
    ^(self new buttonLabel: bString text: tString) open
```

accessing

logoHolder

```
    ""
    "logoHolder := nil"

    ^logoHolder isNil ifTrue: [
        logoHolder := AdventureRandomValueHolder new.
            logoHolder add: MicroslothChairmanBill image.
            logoHolder add: MicroslothInterviewer1 image.
            logoHolder add: MicroslothInterviewer2 image.
            logoHolder add: MicroslothInterviewer3 image.
            logoHolder add: MicroslothParkingLotAttendant image.
            logoHolder add: MicroslothReceptionist image.
            logoHolder add: MicroslothSecurityGuard image]
    ifFalse: [logoHolder]
```

Class: **MicroslothInterviewer**

Superclass: MicroslothActor
Category: Adventure-Microsloth
Class variables: InterviewDisagreementResponses InterviewFailureResponses InterviewSuccessResponses

I'm an MicroslothActor specialized for interviewing duties.

Class Variables:
InterviewDisagreementResponses <AdventureRandomValueHolder> of strings to say when the player disagrees with an assertion I've made.
InterviewFailureResponses <AdventureRandomValueHolder> of strings to say when the player has failed to give an acceptable answer to a question.
InterviewSuccessResponses <AdventureRandomValueHolder> of strings to say when the player has passed the interview.

accessing

interviewDisagreementResponses

```
    ""
    "InterviewDisagreementResponses := nil"

    ^InterviewDisagreementResponses isNil
        ifTrue: [InterviewDisagreementResponses := AdventureRandomValueHolder
                withAll: #(
                    'You don''t have vision!'
                    'I suppose you still code in assembler.'
                    'We don''t like "No" men around here.'
                    'You''ve got a lousy attitude.'
                    'Keep your opinions to yourself!'
                    'Who asked you, anyway?'
                 )]
        ifFalse: [InterviewDisagreementResponses]
```

interviewFailureResponses

```
    ""
    "InterviewFailureResponses := nil"

    ^InterviewFailureResponses isNil
        ifTrue: [InterviewFailureResponses := AdventureRandomValueHolder
                withAll: #(
                    'You''re just not Microsloth material.'
                    'The interview is over. Try Borland.'
                    'Well, I''m a busy man. The interview is over.'
                    'Get some sleep and try us again.'
                    'I''m afraid you''re not going to work out.'
                    'Are you on drugs?'
                    'Sorry--you''re not going to cut it here.'
                    'I''m going to cut the interview short. Sorry.'
                    'I''m afraid I''m going to give you a thumbs-down.'
                 )]
```

```
        ifFalse: [InterviewFailureResponses]
```

interviewSuccessResponses

```
    ""
    "InterviewSuccessResponses := nil"

    ^InterviewSuccessResponses isNil
        ifTrue: [InterviewSuccessResponses := AdventureRandomValueHolder
                withAll: #(
                    'You''re a real forward thinker, Lee.'
                    'As far as I''m concerned, you''re in!'
                    'I''m impressed. Two thumbs up!'
                    'Very impressive.'
                    'You''ll get my recommendation!'
                    'You''re the best candidate so far this month!'
                 )]
        ifFalse: [InterviewSuccessResponses]
```

script actions

praiseAndDispenseToken

```
    self say: self interviewSuccessResponses value.
    self dispenseSlothToken
```

scoldAndEjectPlayerForDisagreement

```
    self say: self interviewDisagreementResponses value.
    self askSecurityGuardToEject: self partner
```

scoldAndEjectPlayerForFailingToAnswer

```
    self say: self interviewFailureResponses value.
    self askSecurityGuardToEject: self partner
```

MetaClass: MicroslothInterviewer class

Class: **MicroslothInterviewer1**

Superclass: MicroslothInterviewer
Category: Adventure-Microsloth

I'm Buddy Suzuki, displaced Wimpdoze/MT evangelist.

initialize-release

initialize

```
    super initialize.
    self
        id: #interviewer1
        name: 'Buddy Suzuki'
        description: 'An Asian male of about 32. He wears tasseled loafers, and a Rolex. He has trouble sitting still. '
        biography: 'Before being "kicked upstairs" to Personnel, Buddy was Microsloth''s Wimpdoze/MT evangelist. He''ll be in Personnel until "things pick up around here, damn!" It was Buddy who coined the slogan "Wimpdoze/MT: Mucho Technology!" '
```

script building

buildScript

```
    | myScript |
    myScript := (AdventureScript named: #myScriptName actor: self)
        add: (AdventureScriptState named: #waitingForPlayer actor: self);
        add: (AdventureScriptState named: #howAreYou actor: self);
        add: (AdventureScriptState named: #whyMicrosloth actor: self);
        add: (AdventureScriptState named: #haveYouHeardOfWimpdozeMT actor: self);
        add: (AdventureScriptState named: #mostPowerfulOSRight actor: self);
        add: (AdventureScriptState named: #year2000MegsAreCheapRight actor: self);
        add: (AdventureScriptState named: #romeNotBuiltInADayRight actor: self);
        add: (AdventureScriptState named: #waitingForPlayerToLeave actor: self);
        add: (AdventureScriptState named: #suspicious actor: self);
        add: (AdventureScriptState named: #hostile actor: self);
        yourself.

"waitingForPlayer"

    self addPlayerAppearanceHandlerTo: (myScript atState: #waitingForPlayer)
        successBlock: nil
        successState: (myScript atState: #howAreYou).

"howAreYou"

    self addQuestionHandlerTo: (myScript atState: #howAreYou)
        questions: #(
            'Hi. You must be Lee. How''re you doing?'
            'Hello Lee. I''m Buddy. How are you today?'
            'Welcome to Microsloth. How''s your experience been so far?'
```

```
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #playerReplied)
                            actor: self
                            matcher: ('*')
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: 'Great! That''s just great.'
                            or: 'So glad to hear it.'
                            or: 'Well, good.']
                    nextState: (myScript atState: #whyMicrosloth)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
        failureState: (myScript atState: #waitingForPlayer).

"whyMicrosloth"

    self addQuestionHandlerTo: (myScript atState: #whyMicrosloth)
        questions: #(
            'Why do want to work at Microsloth?'
            'Tell me why you want to work at Microsloth.'
            'In your own words, tell me why you want to work here.'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #playerMentionedBill)
                            actor: self
                            matcher: (
                                AdventureWordMatcher withAll:
                                    #('Bill' 'Chairman' 'Gates')
                            )
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: 'Of course. Yes, we all came here to work with Bill.'
                            or: 'He IS a visionary, isn''t he?'
                            or: 'Heil Bill!'
                            or: 'It''s quite an opportunity, working with a legend!']
                    nextState: (myScript atState: #haveYouHeardOfWimpdozeMT)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
        failureState: (myScript atState: #waitingForPlayer).

"haveYouHeardOfWimpdozeMT"
```

```
    self addQuestionHandlerTo: (myScript atState: #haveYouHeardOfWimpdozeMT)
        questions: #(
            'You''ve heard of Wimpdoze/MT, the most powerful OS you can get without a doctor''s
prescription, right?'
            'I assume that you''re familiar with Wimpdoze/Mucho Technology. You are, aren''t you?'
            'You''re familiar with Microsloth''s Wimpdoze/MT operating system (code name "VMS"), I
assume?'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureAffirmativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidYes)
                            actor: self
                    )
                    actionBlock: nil
                    nextState: (myScript atState: #mostPowerfulOSRight)
            )
            with: (
                AdventureAnswer
                    cue: (
                        AdventureNegativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidNo)
                            actor: self
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: 'That''s too bad. The last candidate knew all about it.'
                            or: 'Ignorance is not tolerated at ''Sloth.'
                            or: 'Too bad you didn''t do your homework. Bye bye.'.
                        self askSecurityGuardToEject: self partner]
                    nextState: (myScript atState: #waitingForPlayer)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
        failureState: (myScript atState: #waitingForPlayer).

"mostPowerfulOSRight"

    self addQuestionHandlerTo: (myScript atState: #mostPowerfulOSRight)
        questions: #(
            'You know that MT is the most powerful OS ever made, and could blow your head clean off?'
            'It''s the most powerful OS in the Post-Cold-War Free World, right?'
            'I mean, MT is to operating systems what Kraft is to cheddar, right?'
            'It''s the OS of the Gods, right?'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureAffirmativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidYes)
                            actor: self
```

```
                )
                actionBlock: nil
                nextState: (myScript atState: #year2000MegsAreCheapRight)
        )
        with: (
            AdventureAnswer
                cue: (
                    AdventureNegativeSpeechCue
                        machineEvent: (MachineEvent named: #playerSaidNo)
                        actor: self
                )
                actionBlock: [:aScript :anEvent |
                    self scoldAndEjectPlayerForDisagreement]
                nextState: (myScript atState: #waitingForPlayer)
        )
    )
    disappearanceBlock: nil
    disappearanceState: (myScript atState: #waitingForPlayer)
    failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
    failureState: (myScript atState: #waitingForPlayer).

"year2000MegsAreCheapRight"

    self addQuestionHandlerTo: (myScript atState: #year2000MegsAreCheapRight)
        questions: #(
            'In the year 2000, who''s gonna care if it boots in less than 128 megs? Know what I mean?'
            'In the year 2000, who''s gonna care if it needs a 4-gig swap file? Know what I mean?'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureAffirmativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidYes)
                            actor: self
                    )
                    actionBlock: nil
                    nextState: (myScript atState: #romeNotBuiltInADayRight)
            )
            with: (
                AdventureAnswer
                    cue: (
                        AdventureNegativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidNo)
                            actor: self
                    )
                    actionBlock: [:aScript :anEvent |
                        self scoldAndEjectPlayerForDisagreement]
                    nextState: (myScript atState: #waitingForPlayer)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
        failureState: (myScript atState: #waitingForPlayer).
```

```
"romeNotBuiltInADayRight"

    self addQuestionHandlerTo: (myScript atState: #romeNotBuiltInADayRight)
        questions: #(
            'Sure, the only native app''s are "File Manager" and "Clock." Rome wasn''t built in a day,
right?'
            'Sure, the only native app''s are "Program Manager" and "Calculator." Rome wasn''t built in a
day, right?'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureAffirmativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidYes)
                            actor: self
                    )
                    actionBlock: [:aScript :anEvent | self praiseAndDispenseToken]
                    nextState: (myScript atState: #waitingForPlayerToLeave)
            )
            with: (
                AdventureAnswer
                    cue: (
                        AdventureNegativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidNo)
                            actor: self
                    )
                    actionBlock: [:aScript :anEvent |
                        self scoldAndEjectPlayerForDisagreement]
                    nextState: (myScript atState: #waitingForPlayer)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
        failureState: (myScript atState: #waitingForPlayer).

"waitingForPlayerToLeave"
"suspicious"
 "hostile"

    self addPlayerReappearanceHandlerToWaitState:
            (myScript atState: #waitingForPlayerToLeave)
        suspiciousState: (myScript atState: #suspicious)
        hostileState: (myScript atState: #hostile)
        suspiciousResponseHolder: (
            AdventureRandomValueHolder withAll: #(
                'You shouldn''t be back here. You should be with Mr. Walker.'
                'What are you doing here?'
                'You can''t come this way--you should be with your next interviewer.'
            )
        )
        terminationState: (myScript atState: #waitingForPlayer).

    ^myScript
```

MetaClass: MicroslothInterviewer1 class

resources

buildImage

```
^CachedImage on: (Adventure readImageFromBOSSFile: 'male3.bos')
```

Class: **MicroslothInterviewer2**

Superclass: MicroslothInterviewer
Category: Adventure-Microsloth

I'm Mike Gutschmidte-Walker, technical dropout.

initialize-release

initialize

```
	super initialize.
	self
		id: #interviewer2
		name: 'Mike Gutschmidte-Walker'
		description: 'A man of about 35, with a dark beard and mustache. He is wearing Birkenstocks. '
		biography: 'Mike worked for Apple on the ill-fated "Eunice" workstation before joining Microsloth.
He was technical lead on the Wimpdoze 2.0 project, which explains why he''s now in Personnel. Mike still
thinks that the ''286 is "More processor than you''ll ever need!"'
```

script building

buildScript

```
	| myScript |
	myScript := (AdventureScript named: #myScriptName actor: self)
		add: (AdventureScriptState named: #waitingForPlayer actor: self);
		add: (AdventureScriptState named: #howAreYou actor: self);
		add: (AdventureScriptState named: #ninetyHoursAWeekIsOkRight actor: self);
		add: (AdventureScriptState named: #anyOutsideInterests actor: self);
		add: (AdventureScriptState named: #willYouSquanderYourVacation actor: self);
		add: (AdventureScriptState named: #caffeineLevelsTestedOk actor: self);
		add: (AdventureScriptState named: #waitingForPlayerToLeave actor: self);
		add: (AdventureScriptState named: #suspicious actor: self);
		add: (AdventureScriptState named: #hostile actor: self);
		yourself.

"waitingForPlayer"

	self addPlayerAppearanceHandlerTo: (myScript atState: #waitingForPlayer)
		successBlock: nil
		successState: (myScript atState: #howAreYou).

"howAreYou"

	self addQuestionHandlerTo: (myScript atState: #howAreYou)
		questions: #(
			'Hi. Come on in and sit down. How are you?'
			'Hi Lee. I''m Mike. How''s the interview process going?'
			'Hello Lee. How do you like Microsloth so far?'
		)
		answers: (
```

```
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #playerReplied)
                            actor: self
                            matcher: ('*')
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: 'Fabulous. I''m going to ask you a few questions to measure your
compatibility with our corporate culture.'
                            or: 'Great! I thought so! Now, I''m going to test you to see if you can follow
the "Microsloth Way."'
                            or: 'Great. Lets get going. I''m going to ask you a few simple questions to see
if you''re compatible with our corporate culture.']
                    nextState: (myScript atState: #ninetyHoursAWeekIsOkRight)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
        failureState: (myScript atState: #waitingForPlayer).

"ninetyHoursAWeekIsOkRight"

    self addQuestionHandlerTo: (myScript atState: #ninetyHoursAWeekIsOkRight)
        questions: #(
            'I guess that you''re aware that you''ll have to work ninety hours a week?'
            'You''re ok with working ninety hours a week?'
            'You can do ninety hour weeks for extended periods, right?'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureAffirmativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidYes)
                            actor: self
                    )
                    actionBlock: nil
                    nextState: (myScript atState: #anyOutsideInterests)
            )
            with: (
                AdventureAnswer
                    cue: (
                        AdventureNegativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidNo)
                            actor: self
                    )
                    actionBlock: [:aScript :anEvent |
                        self scoldAndEjectPlayerForDisagreement]
                    nextState: (myScript atState: #waitingForPlayer)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
```

```
            failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
            failureState: (myScript atState: #waitingForPlayer).

"anyOutsideInterests"

      self addQuestionHandlerTo: (myScript atState: #anyOutsideInterests)
            questions: #(
                  'You don''t have any outside interests do you?'
                  'You don''t like seeing your family regularly, do you?'
                  'You don''t like the smell of fresh air, or the feeling of sunlight on your skin, do you?'
                  'You don''t aspire to having a life while you''re with us at ''Sloth, do you?'
                  'You don''t mind forgetting the names of your children do you?'
            )
            answers: (
                  Array with: (
                        AdventureAnswer
                              cue: (
                                    AdventureNegativeSpeechCue
                                          machineEvent: (MachineEvent named: #playerSaidNo)
                                          actor: self
                              )
                              actionBlock: nil
                              nextState: (myScript atState: #willYouSquanderYourVacation)
                  )
                  with: (
                        AdventureAnswer
                              cue: (
                                    AdventureAffirmativeSpeechCue
                                          machineEvent: (MachineEvent named: #playerSaidYes)
                                          actor: self
                              )
                              actionBlock: [:aScript :anEvent |
                                    self scoldAndEjectPlayerForDisagreement]
                              nextState: (myScript atState: #waitingForPlayer)
                  )
            )
            disappearanceBlock: nil
            disappearanceState: (myScript atState: #waitingForPlayer)
            failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
            failureState: (myScript atState: #waitingForPlayer).

"willYouSquanderYourVacation"

      self addQuestionHandlerTo: (myScript atState: #willYouSquanderYourVacation)
            questions: #(
                  'You''ll get two weeks vacation, but you''ll be expected to spend it at Comdex helping out at
the booth, ok?'
                  'Newcomers get two weeks vacation, but you''ll be expected to spend it waxing Bill''s car. Is
that ok?'
                  'You get two weeks vacation to start, but you''re expected to volunteer as a fox at the annual
Microsloth "Hunt ''n'' Hounds" celebration. Is that ok by you?'
                  'You''re given two weeks vacation, but we expect you to spend it beta-testing the latest
version of EDLIN for Wimpdoze. Is that all right?'
            )
            answers: (
                  Array with: (
```

```
                    AdventureAnswer
                        cue: (
                            AdventureAffirmativeSpeechCue
                                machineEvent: (MachineEvent named: #playerSaidYes)
                                actor: self
                        )
                        actionBlock: nil
                        nextState: (myScript atState: #caffeineLevelsTestedOk)
                )
                with: (
                    AdventureAnswer
                        cue: (
                            AdventureNegativeSpeechCue
                                machineEvent: (MachineEvent named: #playerSaidNo)
                                actor: self
                        )
                        actionBlock: [:aScript :anEvent |
                            self scoldAndEjectPlayerForDisagreement]
                        nextState: (myScript atState: #waitingForPlayer)
                )
            )
            disappearanceBlock: nil
            disappearanceState: (myScript atState: #waitingForPlayer)
            failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
            failureState: (myScript atState: #waitingForPlayer).

"caffeineLevelsTestedOk"

    self addQuestionHandlerTo: (myScript atState: #caffeineLevelsTestedOk)
        questions: #(
            'We have regular drug testing. Anyone with a caffeine level of less that .10 is fired on the
spot. Do you like coffee?'
            'The bathrooms perform automated drug testing. If your caffeine falls below a certain level,
Security escorts you right to Starbucks. You''re a big cappuccino fan I hope?!'
            'Do you like coffee? Anyone with less than a .10 blood-caffeine level is held down, and forced
to drink a can of "Jolt."'
            'We like our employees hyper-alert, so we constantly monitor caffeine levels via a small
internally-worn blood analyzer. You DO like coffee don''t you?'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureAffirmativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidYes)
                            actor: self
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: 'Very good. If you don''t fit in, you end up in Personnel with me.'
                            or: 'Well done. One wrong answer, and you''d end up in Personnel like me.'
                            or: 'Good answers. One wrong move, and you''d be stuck in Personnel with
me.'.
                        self dispenseSlothToken]
                    nextState: (myScript atState: #waitingForPlayerToLeave)
            )
            with: (
```

```
                AdventureAnswer
                    cue: (
                        AdventureNegativeSpeechCue
                            machineEvent: (MachineEvent named: #playerSaidNo)
                            actor: self
                    )
                    actionBlock: [:aScript :anEvent |
                        self scoldAndEjectPlayerForDisagreement]
                    nextState: (myScript atState: #waitingForPlayer)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
        failureState: (myScript atState: #waitingForPlayer).

"waitingForPlayerToLeave"
"suspicious"
 "hostile"

    self addPlayerReappearanceHandlerToWaitState:
            (myScript atState: #waitingForPlayerToLeave)
        suspiciousState: (myScript atState: #suspicious)
        hostileState: (myScript atState: #hostile)
        suspiciousResponseHolder: (
            AdventureRandomValueHolder withAll: #(
                'Why are you here? You should be with Mr. Spar.'
                'What are you doing back here? You''re expected at the next interview!'
                'There''s no turning back, I''m afraid.'
            )
        )
        terminationState: (myScript atState: #waitingForPlayer).

    ^myScript
```

MetaClass: MicroslothInterviewer2 class

resources

buildImage

```
    ^CachedImage on: (Adventure readImageFromBOSSFile: 'male1.bos')
```

Class: **MicroslothInterviewer3**

Superclass: MicroslothInterviewer
Category: Adventure-Microsloth

I'm Rusty Spar, a political prisoner here in Personnel.

initialize-release

initialize

```
super initialize.
self
    id: #interviewer3
    name: 'Rusty Spar'
    description: 'A 25 year old with red hair and beard. He is wearing an "Alice in Chains" T-shirt. '
    biography: 'Rusty''s first love is music. Rusty and his band "Ctrl-Alt-Del" perform "Kludge Rock" at Microsloth company picnics. Rusty was transferred to Personnel "for his own good" after he refused to boot Wimpdoze 3.1 on his new Pentium "on Religious grounds." Rusty dreams of quitting his job at Microsloth and taking Ctrl-Alt-Del on the road. '
```

script building

buildScript

```
| myScript rightAnswerPraise wrongAnswerScoldings |
myScript := (AdventureScript named: #myScriptName actor: self)
    add: (AdventureScriptState named: #waitingForPlayer actor: self);
    add: (AdventureScriptState named: #howAreYou actor: self);
    add: (AdventureScriptState named: #doYouBootOrDebug actor: self);
    add: (AdventureScriptState named: #threeMonthsOrSixMonths actor: self);
    add: (AdventureScriptState named: #sacrificeQualityOrFunctionality actor: self);
    add: (AdventureScriptState named: #marketingOrEngineeringInCharge actor: self);
    add: (AdventureScriptState named: #whatIsYourFavoriteColor actor: self);
    add: (AdventureScriptState named: #areYouSure actor: self);
    add: (AdventureScriptState named: #waitingForPlayerToLeave actor: self);
    add: (AdventureScriptState named: #suspicious actor: self);
    add: (AdventureScriptState named: #hostile actor: self);
    yourself.

rightAnswerPraise := AdventureRandomValueHolder withAll: #(
    'Excellent!'
    'Great!'
    'Yup.'
    'Perfect!'
    'That''s the spirit!'
    'Well done!'
    'Uh huh.'
    'Right-o.'
).

wrongAnswerScoldings := AdventureRandomValueHolder withAll: #(
```

```
			'Wrong.'
			'No way!'
			'Nope.'
			'Negatory, good buddy.'
			'''fraid not.'
			'Nein!'
			'Not so!'
		).

"waitingForPlayer"

		self addPlayerAppearanceHandlerTo: (myScript atState: #waitingForPlayer)
			successBlock: nil
			successState: (myScript atState: #howAreYou).

"howAreYou"

		self addQuestionHandlerTo: (myScript atState: #howAreYou)
			questions: #(
				'Hi. Have a seat. How''re ya doin''?'
				'Lee! I''m Rusty. How''s the interview going?'
				'Hi Lee. How''s it goin''?'
			)
			answers: (
				Array with: (
					AdventureAnswer
						cue: (
							AdventureSpeechCue
								machineEvent: (MachineEvent named: #playerReplied)
								actor: self
								matcher: ('*')
						)
						actionBlock: [:aScript :anEvent |
							self say: 'Cool. Now, I''m gonna ask you a few questions about programming...'
								or: 'Groovy! Glad to hear it! I''m going to pose a few ethical dilemmas, and
see how you''d solve them.'
								or: 'Bitchin''. Lets get going. I''m going to ask a few questions to get a handle
on your technical philosophy.']
						nextState: (myScript atState: #doYouBootOrDebug)
				)
			)
			disappearanceBlock: nil
			disappearanceState: (myScript atState: #waitingForPlayer)
			failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
			failureState: (myScript atState: #waitingForPlayer).

"doYouBootOrDebug"

		self addQuestionHandlerTo: (myScript atState: #doYouBootOrDebug)
			questions: #(
				'Ok. You''re at Comdex, demoing a new app. Three minutes before the demo, the app
crashes. You''ve seen this before, and you know a reboot clears it up. Just then, the solution hits you!
The fix is a simple edit of a configuration file. My question: do you reboot, or debug?'
			)
			answers: (
				Array with: (
```

```
				AdventureAnswer
					cue: (
						AdventureSpeechCue
							machineEvent: (MachineEvent named: #rightAnswer)
							actor: self
							matcher: (
								AdventureWordMatcher withAll:
									#('*boot*')
							)
					)
					actionBlock: [:aScript :anEvent |
						self say: rightAnswerPraise value]
					nextState: (myScript atState: #threeMonthsOrSixMonths)
			)
			with: (
				AdventureAnswer
					cue: (
						AdventureSpeechCue
							machineEvent: (MachineEvent named: #wrongAnswer)
							actor: self
							matcher: (
								AdventureWordMatcher withAll:
									#('*debug*')
							)
					)
					actionBlock: [:aScript :anEvent |
						self say: wrongAnswerScoldings value.
						self scoldAndEjectPlayerForFailingToAnswer]
					nextState: (myScript atState: #waitingForPlayer)
			)
		)
		disappearanceBlock: nil
		disappearanceState: (myScript atState: #waitingForPlayer)
		failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
		failureState: (myScript atState: #waitingForPlayer).

"threeMonthsOrSixMonths"

	self addQuestionHandlerTo: (myScript atState: #threeMonthsOrSixMonths)
		questions: #(
			'Next. Your manager wants a project done in three months. You know it will take six. What do
you give her as an estimate?'
		)
		answers: (
			Array with: (
				AdventureAnswer
					cue: (
						AdventureSpeechCue
							machineEvent: (MachineEvent named: #rightAnswer)
							actor: self
							matcher: (
								AdventureWordMatcher withAll:
									#('six' '6')
							)
					)
					actionBlock: [:aScript :anEvent |
```

```
                        self say: rightAnswerPraise value]
                    nextState: (myScript atState: #sacrificeQualityOrFunctionality)
            )
            with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #wrongAnswer)
                            actor: self
                            matcher: (
                                AdventureWordMatcher withAll:
                                    #('zero' 'one' 'two' 'three' 'four' 'five' 'seven' 'eight' 'nine' 'ten'
                                        '0' '1' '2' '3' '4' '5' '7' '8' '9' '10')
                            )
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: wrongAnswerScoldings value.
                        self scoldAndEjectPlayerForFailingToAnswer]
                    nextState: (myScript atState: #waitingForPlayer)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
        failureState: (myScript atState: #waitingForPlayer).

"sacrificeQualityOrFunctionality"

    self addQuestionHandlerTo: (myScript atState: #sacrificeQualityOrFunctionality)
        questions: #(
            'All right. You have a fixed deadline that you MUST meet. Generally speaking, is it better to
sacrifice quality or sacrifice functionality to meet it?'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #rightAnswer)
                            actor: self
                            matcher: (
                                AdventureWordMatcher withAll:
                                    #('function*')
                            )
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: rightAnswerPraise value]
                    nextState: (myScript atState: #marketingOrEngineeringInCharge)
            )
            with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #wrongAnswer)
                            actor: self
                            matcher: (
```

```
                        AdventureWordMatcher withAll:
                            #('quality')
                    )
                )
                actionBlock: [:aScript :anEvent |
                    self say: wrongAnswerScoldings value.
                    self scoldAndEjectPlayerForFailingToAnswer]
                nextState: (myScript atState: #waitingForPlayer)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
        failureState: (myScript atState: #waitingForPlayer).

"marketingOrEngineeringInCharge"

    self addQuestionHandlerTo: (myScript atState: #marketingOrEngineeringInCharge)
        questions: #(
            'You''re in a new-product meeting. Who should be in charge, Marketing or Engineering?'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #rightAnswer)
                            actor: self
                            matcher: (
                                AdventureWordMatcher withAll:
                                    #('Market*')
                            )
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: rightAnswerPraise value]
                    nextState: (myScript atState: #whatIsYourFavoriteColor)
            )
            with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #wrongAnswer)
                            actor: self
                            matcher: (
                                AdventureWordMatcher withAll:
                                    #('Engineer*')
                            )
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: wrongAnswerScoldings value.
                        self scoldAndEjectPlayerForFailingToAnswer]
                    nextState: (myScript atState: #waitingForPlayer)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
```

```
		failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
		failureState: (myScript atState: #waitingForPlayer).

"whatIsYourFavoriteColor"

	self addQuestionHandlerTo: (myScript atState: #whatIsYourFavoriteColor)
		questions: #(
			'Finally: What is your favorite color?'
		)
		answers: (
			Array with: (
				AdventureAnswer
					cue: (
						AdventureSpeechCue
							machineEvent: (MachineEvent named: #playerReplied)
							actor: self
							matcher: ('*')
					)
					actionBlock: nil
					nextState: (myScript atState: #areYouSure)
			)
		)
		disappearanceBlock: nil
		disappearanceState: (myScript atState: #waitingForPlayer)
		failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
		failureState: (myScript atState: #waitingForPlayer).

"areYouSure"

	self addQuestionHandlerTo: (myScript atState: #areYouSure)
		questions: #(
			'Are you sure?'
			'Really? Are you sure?'
			'Are you positive?'
		)
		answers: (
			Array with: (
				AdventureAnswer
					cue: (
						AdventureAffirmativeSpeechCue
							machineEvent: (MachineEvent named: #playerSaidYes)
							actor: self
					)
					actionBlock: [:aScript :anEvent | self praiseAndDispenseToken]
					nextState: (myScript atState: #waitingForPlayerToLeave)
			)
			with: (
				AdventureAnswer
					cue: (
						AdventureNegativeSpeechCue
							machineEvent: (MachineEvent named: #playerSaidNo)
							actor: self
					)
					actionBlock: [:aScript :anEvent |
						self say: 'That''s too bad. Self-confidence is important!'
							or: 'Oh no! Without self-confidence, you''ll get eaten alive here at Microsloth.'
```

```
                    or: 'Well, that''s a shame. Self-confidence is essential!'.
                self askSecurityGuardToEject: self partner]
            nextState: (myScript atState: #waitingForPlayer)
        )
    )
    disappearanceBlock: nil
    disappearanceState: (myScript atState: #waitingForPlayer)
    failureBlock: [:aScript :anEvent | self scoldAndEjectPlayerForFailingToAnswer]
    failureState: (myScript atState: #waitingForPlayer).

"waitingForPlayerToLeave"
"suspicious"
 "hostile"

  self addPlayerReappearanceHandlerToWaitState:
        (myScript atState: #waitingForPlayerToLeave)
    suspiciousState: (myScript atState: #suspicious)
    hostileState: (myScript atState: #hostile)
    suspiciousResponseHolder: (
        AdventureRandomValueHolder withAll: #(
            'You should be with Lord Bill!'
            'Lord Bill is expecting you--scoot!'
            'You can''t go back this way. Go and see Lord Bill!'
        )
    )
    terminationState: (myScript atState: #waitingForPlayer).

  ^myScript
```

MetaClass: MicroslothInterviewer3 class

resources

buildImage

```
  ^CachedImage on: (Adventure readImageFromBOSSFile: 'male2.bos')
```

Class: **MicroslothLossDialog**

Superclass: MicroslothInformationDialog
Category: Adventure-Microsloth
Class variables: ButtonLabelHolder TextHolder

I taunt the player when it loses.

Class Variables:
ButtonLabelHolder <AdventureRandomValueHolder> of String’s to label my "accept" button with.
TextHolder <AdventureRandomValueHolder> of String’s to display in my text area.

accessing

buttonLabelHolder

```
""
"ButtonLabelHolder := nil"

^ButtonLabelHolder isNil ifTrue: [
    ButtonLabelHolder := AdventureRandomValueHolder withAll: #(
        ’I Love DOS’
        ’Wimpdoze Rules’
        ’Microsloth Uber Alles’
        ’16 Bits Forever’
        ’Microsloth Means Quality!’
        ’Happiness is a ’’286!’
    )]
ifFalse: [ButtonLabelHolder]
```

textHolder

```
""
"TextHolder := nil"

^TextHolder isNil ifTrue: [
    TextHolder := AdventureRandomValueHolder new.
        TextHolder add: ’You’’ve lost!\(Maybe Borland will take you!)’ withCRs.
        TextHolder add: ’You’’ve lost!\(There’’s always food service.)’ withCRs.
        TextHolder add: ’You’’ve lost!\(Get an MBA.)’ withCRs.
        TextHolder add: ’You’’ve lost!\(There’’s always marketing...)’ withCRs.
        TextHolder add: ’You’’ve lost!\(Maybe you should try Retail Engineering.)’
            withCRs.
        TextHolder add: ’You’’ve lost!\(Dang! 10 years of college... for nothing!)’
            withCRs.
        TextHolder add: ’You’’ve lost!\(Too much LDS in the 60’’s huh?)’ withCRs.
        TextHolder add: ’You’’ve lost!\(You’’re no Chairman Bill!)’ withCRs]
ifFalse: [TextHolder]
```

initialize-release

initialize

```
super initialize.
```

```
self buttonLabel: self buttonLabelHolder value text: self textHolder value
```

MetaClass: MicroslothLossDialog class

Class: **MicroslothParkingLotAttendant**

Superclass: MicroslothActor
Category: Adventure-Microsloth

I’m Carl A. Fenner, recent Microsloth hire.

initialize-release

bmw990iKey

```
^AdventureKey
    id: #bmw990iKey
    name: 'Car Key'
    description: 'A silver key with the BMW logo in the middle. '
```

initialize

```
super initialize.
self
    id: #parkingLotAttendant
    name: 'Carl A. Fenner'
    description: 'Today is Carl''s first day at Microsloth. Even so, he''s starting to develop a tan from long hot hours in the Microsloth parking lot. He is jamming to "Alice in Chains" on his portable CD player. '
    biography: 'A 19-year Sophomore on full scholarship at MIT, Carl is here at Microsloth through the Work-Study program. When he was promised that he would "work side-by-side with Chairman Bill," he didn''t realize that meant he''d spend all day in the parking lot watching Bill''s 990i "so nobody dings it." '.
self pocket at: #bmw990iKey put: self bmw990iKey
```

script building

buildScript

```
| myScript  greetings |
myScript := (AdventureScript named: #parkingAttendantScript actor: self)
    add: (AdventureScriptState named: #waitingForPlayer actor: self);
    add: (AdventureScriptState named: #pondering actor: self);
    add: (AdventureScriptState named: #waitingForPlayerToImpersonateBill actor: self);
    add: (AdventureScriptState named: #waitingForPlayerToLeave actor: self);
    add: (AdventureScriptState named: #suspicious actor: self);
    add: (AdventureScriptState named: #hostile actor: self);
    yourself.

"waitingForPlayer"

greetings := AdventureRandomValueHolder withAll: #(
    'Hi, I''m Carl. I''ve been watching your car all day. Phillipe Kahn tried to soap the windows, but I called Security.'
    'Chairman Bill! I can''t tell you how great it is to meet you! I''ve always admired your work on MONKEY.BAS!'
    'Chairman Bill! It''s great to finally meet you. I''ve been polishing the muffler with my tongue like you asked!'
    'Hello! I''m Carl Fenner. I saw you on "Eye to Eye." Man, Connie Chung must still walk with a
```

```
limp!'
		'Chairman Bill?! A car full of anti-trust lawyers stopped for directions. I sent ''em to Portland.'
	).

	self addPlayerPresenceHandlerTo: (myScript atState: #waitingForPlayer)
		successBlock: [:aScript :anEvent |
			self say: greetings value]
		successState: (myScript atState: #pondering).

"pondering"

	self addWaitHandlerTo: (myScript atState: #pondering)
		successBlock: nil
		successState: (myScript atState: #waitingForPlayerToImpersonateBill)
		disappearanceBlock: nil
		disappearanceState: (myScript atState: #waitingForPlayer)
		seconds: 4.

"waitingForPlayerToImpersonateBill"

	self addQuestionHandlerTo: (myScript atState: #waitingForPlayerToImpersonateBill)
		questions: #(
			'Say... You ARE Chairman Bill aren''t you?'
			'Is it really you Chairman Bill?!!'
			'You are Him, aren''t you?'
			'You look taller in person. You really are Bill, right?'
		)
		answers: (
			Array with: (
				AdventureAnswer
					cue: (
						AdventureAffirmativeSpeechCue
							machineEvent: (MachineEvent named: #playerClaimsToBeBill)
							actor: self
					)
					actionBlock: [:aScript :anEvent |
						self say: 'Ok. Here is your key, Chairman. This is a nice car. Love the machine
guns...'
							or: 'Here is your key. Nice wheels, by the way. I''ve never seen a BMW with
chrome headers before.'
							or: 'Here''s your key. By the way, I love the "Microsloth Tribe" decal in the
back window.'.
						(self pocket removeKey: #bmw990iKey) container: self partner]
					nextState: (myScript atState: #waitingForPlayerToLeave)
			)
			with: (
				AdventureAnswer
					cue: (
						AdventureNegativeSpeechCue
							machineEvent: (MachineEvent named: #playerDenysBeingBill)
							actor: self
					)
					actionBlock: [:aScript :anEvent |
						self say: 'So you admit it! I''m calling Security!'
							or: 'I knew it all along! I''m calling Security!'
							or: 'Your people will pay for this!'.
```

```
                        self askSecurityGuardToTerminate: self partner]
                    nextState: (myScript atState: #waitingForPlayer)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent |
            self say: 'I''m calling Security!'
                or: 'That''s it! You''re outta here!'
                or: 'It''s time to meet Spike.'.
            self askSecurityGuardToTerminate: self partner]
        failureState: (myScript atState: #waitingForPlayer).

"waitingForPlayerToLeave"
"suspicious"
 "hostile"

    self addPlayerReappearanceHandlerToWaitState:
            (myScript atState: #waitingForPlayerToLeave)
        suspiciousState: (myScript atState: #suspicious)
        hostileState: (myScript atState: #hostile)
        suspiciousResponseHolder: (
            AdventureRandomValueHolder withAll: #(
                'Hey! I don''t think you''re Chairman Bill after all! Show me your ID.'
                'You''re not Bill, are you? Show me some ID.'
                'You''re.. you''re an impostor! You''re not Chairman Bill at all.'
            )
        )
        terminationState: (myScript atState: #waitingForPlayer).

    ^myScript
```

MetaClass: MicroslothParkingLotAttendant class

resources

buildImage

```
    ^CachedImage on: (Adventure readImageFromBOSSFile: 'carl.bos')
```

Class: **MicroslothReceptionist**

Superclass: MicroslothActor
Category: Adventure-Microsloth

I am Microsloth Corp's receptionist.

initialize-release

initialize

```
	super initialize.
	self
		id: #receptionist
		name: 'Tiffany Johansen'
		description: 'An attractive young woman with golden hair (courtesy Deutsche Kosmetische Produkte Fabrik GmbH.) She is wearing Revlon''s newest brand of lipstick "Hyperfection" in "Really Really Red!!!"'
		biography: 'A beautiful 20-year-old of indeterminate hair-color, and regular reader of "Cosmo," Tiffany is Microsloth Corporation''s first line of defense. She thinks Chairman Bill is "Cute." '
```

script building

buildScript

```
	| myScript |
	myScript := (AdventureScript named: #receptionistScript actor: self)
		add: (AdventureScriptState named: #waitingForPlayer actor: self);
		add: (AdventureScriptState named: #waitingForPlayerToMentionJob actor: self);
		add: (AdventureScriptState named: #waitingForPlayerToStateCorrectName actor: self);
		add: (AdventureScriptState named: #waitingForPlayerToLeave actor: self);
		add: (AdventureScriptState named: #suspicious actor: self);
		add: (AdventureScriptState named: #hostile actor: self);
		yourself.

"waitingForPlayer"

	self addPlayerAppearanceHandlerTo: (myScript atState: #waitingForPlayer)
		successBlock: nil
		successState: (myScript atState: #waitingForPlayerToMentionJob).

"waitingForPlayerToMentionJob"

	self addQuestionHandlerTo: (myScript atState: #waitingForPlayerToMentionJob)
		questions: #(
			'We''re having a great day at Microsloth Corporation! My name is Tiffany. I''m anxious to help you in any way I can!'
			'Welcome to Microsloth Corporation, home of DOS: the Non-Reentrant Single-User Interrupt Handler! My name is Tiffany.  How can I help you?'
			'Welcome to Microsloth Corporation, home of Wimpdoze 3.1: the 16-Bit Terminal Driver on Crack! My name is Tiffany. What can I do for you today?'
		)
		answers: (
```

```
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #playerMentionedJob)
                            actor: self
                            matcher: (
                                AdventureWordMatcher withAll:
                                    #('hire' 'job' 'interview' 'employ*')
                            )
                    )
                    actionBlock: nil
                    nextState: (myScript atState: #waitingForPlayerToStateCorrectName)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent |
            self say: 'I''m afraid you''re going to have to leave. Spike!!!'.
            self askSecurityGuardToEject: self partner]
        failureState: (myScript atState: #waitingForPlayer).

"waitingForPlayerToStateCorrectName"

    self addQuestionHandlerTo: (myScript atState: #waitingForPlayerToStateCorrectName)
        questions: #(
            'If you''ll just give me your name, I''ll get you going.'
            'Please give me your name.'
        )
        answers: (
            Array with: (
                AdventureAnswer
                    cue: (
                        AdventureSpeechCue
                            machineEvent: (MachineEvent named: #playerStatedItsName)
                            actor: self
                            matcher: (
                                AdventureWordMatcher withAll:
                                    #('Lee' 'Player')
                            )
                    )
                    actionBlock: [:aScript :anEvent |
                        self say: 'Ah yes, you''ll be seeing Mr. Suzuki in Interview Room 1.'.
                        self dispenseSlothToken]
                    nextState: (myScript atState: #waitingForPlayerToLeave)
            )
        )
        disappearanceBlock: nil
        disappearanceState: (myScript atState: #waitingForPlayer)
        failureBlock: [:aScript :anEvent |
            self say: 'Spike! Fetch!!!'
                or: 'Spike! Kill!'
                or: 'Spike! Have this person removed from Our presence.'.
            self askSecurityGuardToEject: self partner]
        failureState: (myScript atState: #waitingForPlayer).
```

```
"waitingForPlayerToLeave"
"suspicious"
"hostile"

	self addPlayerReappearanceHandlerToWaitState:
			(myScript atState: #waitingForPlayerToLeave)
		suspiciousState: (myScript atState: #suspicious)
		hostileState: (myScript atState: #hostile)
		suspiciousResponseHolder: (
			AdventureRandomValueHolder withAll: #(
				'What are you doing here? You should be interviewing with Mr. Suzuki.'
				'You shouldn''t be back here. You should be with Mr. Suzuki.'
				'You can''t come this way--you should be with your next interviewer.'
			)
		)
		terminationState: (myScript atState: #waitingForPlayer).

	^myScript
```

MetaClass: MicroslothReceptionist class

resources

buildImage

```
	^CachedImage on: (Adventure readImageFromBOSSFile: 'blonde.bos')
```

Class: **MicroslothSecurityGuard**

Superclass: MicroslothActor
Category: Adventure-Microsloth
Instance variables: threats

I’m Spike, Microsloth’s enforcer.

Instance Variables:
threats <AdventureRandomValueHolder> of String’s to threaten players with just before they are ejected or terminated.

initialize-release

initialize

```
super initialize.
self
    id: #securityGuard;
    name: 'Marion "Spike" Swindell';
    description: 'A tall, blonde security guard with short cropped hair. He is wearing a pair of Perp-Spotter (C) mirrored sunglasses, standard issue at Microsloth. ';
    biography: 'As Microsloth''s best-groomed security guard, Spike guards the reception area. A consummate professional, Spike holds a torch for Tiffany, but still calls her "Miss Johansen." Spike is ex-Coast Guard. He was dishonorably discharged for torpedoing a cruise ship. Spike is determined to keep his nose clean at Microsloth. '
```

manipulate players

eject: aPlayer for: anActor

```
self
    perform: [:player :actor | player container: (self adventure componentAt: #parkingLot)]
    player: aPlayer
    actor: anActor
```

inhabit: aContainer

```
self container ~~ aContainer ifTrue: [self container: aContainer]
```

perform: aBlock player: aPlayer actor: anActor

```
| saved |
saved := self container.
self inhabit: anActor container.
(Delay forMilliseconds: 2000) wait.
self threaten.
(Delay forMilliseconds: 2000) wait.
aBlock value: aPlayer value: anActor.
self inhabit: saved
```

terminate: aPlayer for: anActor

```
self
    perform: [:player :actor | player lose]
    player: aPlayer
```

```
        actor: anActor

threaten
    self say: self threats value
```

accessing

```
threats
    ^threats isNil
        ifTrue: [threats :=
            AdventureRandomValueHolder withAll: #(
                'Dead or alive, you''re coming with me.'
                'You''ve gotta ask yourself one question: "Do I feel lucky?"'
                'You''re not gonna believe what happens next, even while it''s happenin''.'
                'Hasta La Vista, Baby.'
                'You know, if I wanted to, I could crush your head like a little walnut!'
                'It''s people like you, what cause unrest...'
                'Your move, creep.'
                'So, say good-bye to all of THIS, and hello... to oblivion!'
             )]
        ifFalse: [threats]

threats: aRandomValueHolder
    threats := aRandomValueHolder
```

script building

```
buildScript

    | myScript  |
    myScript := (AdventureScript named: #myScriptName actor: self)
        add: (AdventureScriptState named: #waitingForPlayer actor: self);
        add: (AdventureScriptState named: #waitingForPlayerToStealBriefcase actor: self);
        yourself.

"waitingForPlayer"

    self addPlayerAppearanceHandlerTo: (myScript atState: #waitingForPlayer)
        successBlock: nil
        successState: (myScript atState: #waitingForPlayerToStealBriefcase).

"waitingForPlayerToStealBriefcase"

    "Begin cues in order"
    self addPlayerDisappearanceHandlerTo:
            (myScript atState: #waitingForPlayerToStealBriefcase)
        successBlock: nil
        successState: (myScript atState: #waitingForPlayer).

    (myScript atState: #waitingForPlayerToStealBriefcase)
        addCue: (
            AdventureComponentCue
                machineEvent: (MachineEvent named: #briefcaseHasDisappeared)
                actor: self
                eventType: #disappearanceEvent
```

```
            componentId: #briefcase
    ).
"End cues in order"

(myScript atState: #waitingForPlayerToStealBriefcase)
    add: (
        (MachineTransitions forEvent: #briefcaseHasDisappeared)
            add: (
                MachineTransition new
                    actionBlock: [:aScript :anEvent |
                        self say: ’I saw that!’
                            or: ’We don’’t tolerate COMMON thievery at Microsloth!’
                            or: ’That briefcase doesn’’t belong to you!’
                            or: ’I didn’’t see you come in with that briefcase!’.
                        self terminate: self partner for: self];
                    nextState: (myScript atState: #waitingForPlayer)
            )
    ).

^myScript
```

MetaClass: MicroslothSecurityGuard class

resources

buildImage

```
^CachedImage on: (Adventure readImageFromBOSSFile: ’guard.bos’)
```

Class: **MicroslothTelevision**

Superclass: AdventureRobot
Category: Adventure-Microsloth
Instance variables: isOn currentLine dialog isWorking

I'm a television that spews a stream of dialog. I can be turned on and off. At then end of my dialog, I generate a UEA error.

Instance Variables:
isOn <Boolean> If true, I'm on and saying dialog.
currentLine <Integer> index of the current line of dialog.
isWorking <Boolean> If true, I'm working.
dialog <Collection> of String's that make up my dialog.

behavior

behaviorBlock

```
^[
[(Delay forMilliseconds: 3000) wait.
self executeTurn: [(self isWorking and: [self isOn and: [self playerIsPresent]])
        ifTrue: [self sayNextLineOfDialog]]] repeat]
```

playerIsPresent

```
self container contents do: [:aThing | (aThing isKindOf: AdventurePlayer)
        ifTrue: [^true]].
^false
```

sayNextLineOfDialog

```
self say: (self dialog at: self currentLine).
self currentLine: self currentLine + 1.
self currentLine > self dialog size
    ifTrue:
        [self isWorking: false.
        MicroslothWimpdozeUAEDialog open]
```

accessing

currentLine

```
^currentLine
```

currentLine: anInteger

```
currentLine := anInteger
```

description

```
^super description , ' It is ' , (self isOn
        ifTrue: ['on']
        ifFalse: ['off']) , (self isWorking
        ifTrue: [' and working. ']
        ifFalse: [' and broken. '])
```

dialog
^dialog

dialog: aCollection
dialog := aCollection

isOn
^isOn

isOn: aBoolean
isOn := aBoolean.
self changed: #description

isWorking
^isWorking

isWorking: aBoolean
isWorking := aBoolean.
self changed: #description

menu items

addMenuItemsFor: aPlayer to: aCollection
```
aCollection add: (self isOn
        ifTrue: [AdventureMenuItem label: ’off’ block: [self isOn: false]]
        ifFalse: [AdventureMenuItem label: ’on’ block: [self isOn: true]]).
super addMenuItemsFor: aPlayer to: aCollection
```

initialize-release

initialize
super initialize.
self name: ’Television’.
self description: ’A 27" Sony color television, connected to the Microsloth Cable Network. A sticker reads "Wimpdoze Inside!" ’.
self isMoveable: false

withDialog: aCollection
self dialog: aCollection.
self currentLine: 1.
self isOn: true.
self isWorking: true

MetaClass: MicroslothTelevision class

instance creation

withDialog: aCollection
^self new withDialog: aCollection

Class: **MicroslothToken**

Superclass: AdventureToken
Category: Adventure-Microsloth

I'm the coin of the realm at Microsloth.

initialize-release

initialize

```
	super initialize.
	self
		shape: #slothToken;
		value: 1;
		name: ’Sloth Token’;
		description: ’Coin of the realm at Microsloth. The obverse side features a smiling image of Chairman Bill and the words "In Sloth We Trust". The reverse pictures a marmot rampant, and the motto "Death Before Documentation." ’
```

MetaClass: MicroslothToken class

resources

buildImage

```
	^CachedImage on: (Adventure readImageFromBOSSFile: ’sloth.bos’)
```

Class: **MicroslothTokenReturnMechanism**

Superclass: AdventureProperty
Category: Adventure-Microsloth
Instance variables: tokens

I am a very simple property. Generally, I'm initialized containing 0 or more tokens. When my "token return button" is pressed, I dump the tokens into my owner's environment.

Instance Variables:
tokens <OrderedCollection> My collection of tokens.

menu items

addMenuItemsFor: aPlayer to: aCollection onBehalfOf: aThing actingAs: roleSymbol
roleSymbol = #owner ifTrue: [aCollection add: (AdventureMenuItem label: 'press coin return' block: [self tokens size timesRepeat: [self tokens removeFirst container: self owner container]])]

initialize-release

initialize
```
super initialize.
self description: 'token return'
```

accessing

tokens
```
^tokens isNil
    ifTrue: [tokens := OrderedCollection new]
    ifFalse: [tokens]
```

tokens: aValue
```
tokens := aValue
```

forwarding

performAndForward: aSymbol performFirst: aBoolean
```
self
    performAndForward: aSymbol
    performFirst: aBoolean
    to: self tokens
```

performAndForward: aSymbol with: aParameter performFirst: aBoolean
```
self
    performAndForward: aSymbol
    with: aParameter
    performFirst: aBoolean
    to: self tokens
```

MetaClass: MicroslothTokenReturnMechanism class

Class: **MicroslothVideoGame**

Superclass: AdventureThing
Category: Adventure-Microsloth
Instance variables: isWorking

I'm a video game. When you deposit a sloth into me, I display a humorous dialog if I'm working. If I'm broken, I just eat your sloth and taunt you.

Instance Variables:
isWorking <Boolean> flag indicating my state. This is read by my #videoGameMechanism. Changing it updates my #description.

initialize-release

initialize
```
    super initialize.
    self name: 'Video Game'.
    self description: 'Microsloth''s latest arcade offering: "Nuke ''Em!". It runs on Wimpdoze for Coin-Operated Video Games 3.1. '.
    self propertyAt: #videoGameMechanism put: MicroslothVideoGameMechanism new.
    self propertyAt: #tokenReturnMechanism put: MicroslothTokenReturnMechanism new.
    self isMoveable: false
```

accessing

description
```
    ^self isWorking ifFalse: [super description , 'A sign reads "Out Of Order." ']
        ifTrue: [super description]
```

isWorking
```
    ^isWorking isNil
        ifTrue: [isWorking := true]
        ifFalse: [isWorking]
```

isWorking: aBoolean
```
    isWorking := aBoolean.
    self changed: #description
```

MetaClass: MicroslothVideoGame class

resources

buildImage
```
    ^CachedImage on: (Adventure readImageFromBOSSFile: 'game.bos')
```

instance creation

withTokens: anInteger

```
| game |
game := self new.
anInteger timesRepeat: [(game propertyAt: #tokenReturnMechanism) tokens add: MicroslothToken new].
^game
```

Class: **MicroslothVideoGameMechanism**

Superclass: AdventureTokenOperatedMechanism
Category: Adventure-Microsloth

I'm the guts of a MicroslothVideoGame. When the user drops in a sloth, I display a funny dialog, or just eat the sloth, depending on the state of my owner (the MicroslothVideoGame.)

actuating

brokenActuation

```
    self owner
        say: 'I''m broken, but thanks for the sloth anyway.'
        or: 'Can''t you read? I''m broken.'
        or: 'Read the sign, Einstein.'
        or: 'I''m broken. Sorry.'
```

displayCursorNoise

```
    | cursors random |
    cursors := AdventureRandomValueHolder new.
    random := Random new.
    cursors
        add: Cursor thumbsDown;
        add: Cursor crossHair;
        add: Cursor quickGarbage;
        add: Cursor crossHair;
        add: Cursor marker;
        add: Cursor square;
        add: Cursor scroll;
        add: Cursor caret;
        add: Cursor corner;
        add: Cursor execute;
        add: Cursor read;
        add: Cursor thumbsUp;
        add: Cursor up;
        add: Cursor write;
        add: Cursor wait.
    cursors do: [:aCursor | aCursor showWhile: [(Delay forMilliseconds: (random next * 100) asInteger)
wait]]
```

workingActuation

```
    self displayCursorNoise.
    MicroslothWimpdozeUAEDialog open
```

initialize-release

initialize

```
    super initialize.
    self shape: #slothToken.
    self actuateBlock: [:aPlayer | self owner isWorking
            ifTrue: [self workingActuation]
```

```
        ifFalse: [self brokenActuation]]
```

MetaClass: MicroslothVideoGameMechanism class

Class: **MicroslothWimpdozeUAEDialog**

Superclass: SimpleDialog
Category: Adventure-Microsloth
Class variables: DialogBoxTextHolder

I display a simulated Windows 3.x UAE dialog, followed by a scary glimpse of a DOS command line.

Class Variables:
DialogBoxTextHolder <AdventureRandomValueHolder> on the String or Text that is displayed to explain the nature of the error.

aspects

dialogBoxText

```
^self dialogBoxTextHolder value
```

interface opening

preBuildWith: aBuilder

```
aBuilder policy: Win3LookPolicy new
```

actions

exitToDOS

```
MicroslothDOSCommandLine openForMilliseconds: nil.
self accept value: true
```

accessing

dialogBoxTextHolder

```
""
"DialogBoxTextHolder := nil"

^DialogBoxTextHolder isNil
    ifTrue: [DialogBoxTextHolder := (AdventureRandomValueHolder new)
        add: ’Untested Application
Error\CS:IP=0F:4FC0\FLG:0013\SS:SP=0F:5413\WIMPDOZE.DLL\Your application has entered an
indeterminate state.\Critical system data structures may be corr‘$^FF:!~__%^^^^^,K45’’>#4\File system
cache may have been inadvertently modified.\Interrupt vectors may have been copied over serial input
buffers.\Stack may have over-written CMOS RAM.\Hitchhikers may be escaping convicts.’ withCRs;
        add: ’Unbelievable Application
Error\CS:IP=0F:34D0\FLG:0035\SS:SP=0E:0666\WIMPDOZE.DLL\Your application has detected an
undetectable error.\File system may be cross-linked with Excel spreadsheets.\Modem setup data may
have replaced boot block.\EXE files may have been XORed with video BIOS ROMs.\Compressed
partitions may have undergone explosive decompression.\Objects in monitor are closer than they appear.’
withCRs;
        add: ’Usual Application
Error\CS:IP=01:4450\FLG:0012\SS:SP=04:5563\WIMPDOZE.DLL\Your application is dead, Jim.\Printer
boot ROMs may have overwritten monitor firmware.\Mouse button 2 may be connected to alternating
current.\FAT entries may have been converted to EBCDIC.\Superprompt critical power excursion
```

```
detected.\Drive C: format in progress.' withCRs;
			yourself]
		ifFalse: [DialogBoxTextHolder]
```

MetaClass: MicroslothWimpdozeUAEDialog class

interface specs

windowSpec

```
	"UIPainter new openOnClass: self andSelector: #windowSpec"

	^#(#FullSpec #window: #(#WindowSpec #label: 'MicroslothWimpdozeUAEDialog' #min: #(#Point 50 50 ) #bounds: #(#Rectangle 379 306 971 601 ) ) #component: #(#SpecCollection #collection: #(#(#ActionButtonSpec #layout: #(#Rectangle 16 248 96 280 ) #model: #exitToDOS #label: 'Abort' #defaultable: true ) #(#ActionButtonSpec #layout: #(#Rectangle 112 248 192 280 ) #model: #exitToDOS #label: 'Retry' #defaultable: true ) #(#ActionButtonSpec #layout: #(#Rectangle 208 248 288 280 ) #model: #exitToDOS #label: 'Fail' #defaultable: true ) #(#ActionButtonSpec #layout: #(#Rectangle 496 248 576 280 ) #model: #exitToDOS #label: 'Cancel' #defaultable: true ) #(#ActionButtonSpec #layout: #(#Rectangle 400 248 480 280 ) #model: #exitToDOS #label: 'Ok' #isDefault: true #defaultable: true ) #(#ActionButtonSpec #layout: #(#Rectangle 304 248 384 280 ) #model: #exitToDOS #label: 'Ignore' #defaultable: true ) #(#TextEditorSpec #layout: #(#Rectangle 16 16 576 240 ) #flags: 4 #model: #dialogBoxText #style: #system #isReadOnly: true ) ) ) )
```

Class: **MicroslothWinDialog**

Superclass: SimpleDialog
Category: Adventure-Microsloth
Class variables: TextHolder

I congratulate the player when it wins.

Class Variables:
TextHolder <AdventureRandomValueHolder> of String's to display in my text box.

aspects

text

```
^self textHolder value
```

accessing

textHolder

```
""
"TextHolder := nil"

^TextHolder isNil
    ifTrue:
        [TextHolder := AdventureRandomValueHolder new.
        TextHolder add:
            'You''ve won!\(Now about that firstborn child...)' withCRs.
        TextHolder add:
            'You''ve won!\(Captain, you''ve heard of Col. Phillipe E. Kurtz? Operations officer,
Borland Special Forces?)' withCRs.
        TextHolder add:
            'You''ve won!\(Report to the parking lot next Monday.)' withCRs.
        TextHolder add:
            'You''ve won!\(Leave your soul with the receptionist on your way out.)' withCRs]
    ifFalse: [TextHolder]
```

MetaClass: MicroslothWinDialog class

interface specs

windowSpec

```
"UIPainter new openOnClass: self andSelector: #windowSpec"

^#(#FullSpec #window: #(#WindowSpec #label: 'Microsloth Win Dialog' #min: #(#Point 20 20 )
#bounds: #(#Rectangle 395 324 775 535 ) ) #component: #(#SpecCollection #collection:
#(#(#ActionButtonSpec #layout: #(#Rectangle 72 144 168 200 ) #model: #cancel #label: #sign
#hasCharacterOrientedLabel: false #defaultable: true ) #(#LabelSpec #layout: #(#Point 240 8 ) #label:
```

#contract #hasCharacterOrientedLabel: false) #(#LabelSpec #layout: #(#Point 8 8) #label: #satan #hasCharacterOrientedLabel: false) #(#TextEditorSpec #layout: #(#Rectangle 80 8 240 144) #flags: 4 #model: #text #isReadOnly: true))))

resources

contract
```
^CachedImage on: (Adventure readImageFromBOSSFile: 'contract.bos')
```

satan
```
^CachedImage on: (Adventure readImageFromBOSSFile: 'satan.bos')
```

sign
```
^CachedImage on: (Adventure readImageFromBOSSFile: 'sign.bos')
```