Chapter 1

Building a mini actor model

1.1 Making a nicer ProtoObject

```
ProtoObject subclass: #ATMinimalLearningObject
    instanceVariableNames: ''
    classVariableNames: ''
    package: 'MiniActalk2'
```

```
ATMinimalLearningObject >>  doesNotUnderstand: aMessage

    | sel |
    sel := aMessage selector.
    Transcript show: 'Does not understand ',  sel printString ; cr.
    ^ (Object includesSelector: sel)
        ifTrue: [
                self addFromObjectSelector: sel.
                aMessage sendTo: self ]
        ifFalse: [ super doesNotUnderstand: aMessage ].
```

```
ATMinimalLearningObject >>  addFromObjectSelector: aSelector

    | method |
    method := OpalCompiler new
                    class: self class;
                    source: (Object sourceCodeAt: aSelector);
                    compile.
    self class addSelector: aSelector withMethod: method.
```

1.2 Actor

```
ATMinimalLearningObject subclass: #ATActor
    instanceVariableNames: 'mailbox behavior'
    classVariableNames: ''
    package: 'MiniActalk2'
```

```
ATActor >> initialize
    mailbox := SharedQueue new.
```

```
ATActor >> initializeBehavior: aBehavior

    behavior := aBehavior.
    behavior initializeSelf: self
```

```
ATActor >> aSynchronousSend: aMessage

    mailbox nextPut: aMessage
```

```
ATActor class >> behavior: aBehavior
    ^ self new initializeBehavior: aBehavior
```

1.3 Actor Behavior

```
Object subclass: #ATActorBehavior
    instanceVariableNames: 'aself process'
    classVariableNames: ''
    package: 'MiniActalk2'
```

```
initializeASelf: anActor
    aself := anActor.
    self setProcess.
```

```
setProcess
    process := [ [ true ] whileTrue: [ self acceptNextMessage]].
    process fork
```

```
acceptNextMessage
    self acceptMessage: aself mailbox next
```

```
acceptMessage: aMessage
    ^ aMessage sendTo: self
```

```
asynchronousSend: aMessage
    mailbox nextPut: aMessage
```

```
doesNotUnderstand: aMessage
    self asynchronousSend: aMessage
```

1.4 Example: a Counter

```
ATActorBehavior subclass: #ATCounter
    instanceVariableNames: 'contents'
    classVariableNames: ''
    package: 'MiniActalk2-Example'
```

```
increment
    Transcript show: 'increment';cr.
    contents := contents + 1
```

```
reset
    Transcript show: 'reset';cr.
    contents := 0
```

```
| atCounter |
atCounter := ATCounter new actor.
atCounter
    asynchronousSend: (Message selector: #reset);
    asynchronousSend: (Message selector: #increment).
```

```
| atCounter |
atCounter := ATCounter new actor.
atCounter  reset; increment.
```

With Printer

```
ATActorBehavior subclass: #ATPrinter
    instanceVariableNames: ''
    classVariableNames: ''
    package: 'MiniActalk2-Example'
```

```
ATPrinter >> reply: value
    Transcript show: '> ', value printString; cr
```

```
ATCounter >> valueTo: replyDestination
    replyDestination reply: contents
```

```
| atCounter atPrinter |
atCounter := ATCounter new actor.
atPrinter := ATPrinter new actor.
atCounter  reset; increment; valueTo: atPrinter
```