# 11. Exemplary Solutions: Bytecode

## Exercise 11.1:

```
| irMethod aCompiledMethod res |
irMethod:= IRBuilder new
    numRargs:  1;
    addTemps: #(self);
    pushLiteral: 3;
    pushLiteral: 2;
    send: #+;
    pushLiteral: 2;
    send: #*;
    send: #factorial;
    returnTop;
    ir.

aCompiledMethod  := irMethod compiledMethod.
res := aCompiledMethod valueWithReceiver:  nil arguments: #().
self assert: res = 3628800.
```

## Exercise 11.2:

```
| irMethod aCompiledMethod res |
irMethod:= IRBuilder new
    numRargs:  3;
    addTemps: #(self a b);      "receiver and args"
    pushTemp: #a;
    pushTemp: #b;
    send: #+;
    send: #factorial;
    returnTop;
    ir.

aCompiledMethod  := irMethod compiledMethod.
res := aCompiledMethod valueWithReceiver:  nil arguments: #(3 4).
self assert: res = 5040.
```

## Exercise 11.3:

```
| irMethod aCompiledMethod res |
irMethod:= IRBuilder new
    numRargs:  1;
    addTemps: #(self);      "receiver and args"
    pushInstVar: 1;
    pushInstVar: 2;
```

```
    send: #+;
    returnTop;
    ir.

aCompiledMethod  := irMethod compiledMethod.
res := aCompiledMethod valueWithReceiver: 3@4 arguments: #().
self assert: res = 7.
```

### Exercise 11.4:

WIth closure specific bytecodes:

```
13 <8A 01> push: (Array new: 1)
15 <69> popIntoTemp: 1
16 <75> pushConstant: 0
17 <8E 00 01> popIntoTemp: 0 inVectorAt: 1
20 <11> pushTemp: 1
21 <8F 10 00 0A> closureNumCopied: 1 numArgs: 0 bytes 25 to 34
25  <10> pushTemp: 0
26  <8F 10 00 04> closureNumCopied: 1 numArgs: 0 bytes 30 to 33
30  <8C 00 00> pushTemp: 0 inVectorAt: 0
33  <7D> blockReturn
34  <7D> blockReturn
35 <68> popIntoTemp: 0
36 <76> pushConstant: 1
37 <8E 00 01> popIntoTemp: 0 inVectorAt: 1
40 <10> pushTemp: 0
41 <C9> send: value
42 <68> popIntoTemp: 0
43 <77> pushConstant: 2
44 <8E 00 01> popIntoTemp: 0 inVectorAt: 1
47 <10> pushTemp: 0
48 <C9> send: value
49 <7C> returnTop
```

WIthout closure specific bytecodes:

```
13 <75> pushConstant: 0
14 <68> popIntoTemp: 0
15 <89> pushThisContext:
16 <75> pushConstant: 0
17 <C8> send: blockCopy:
18 <A4 08> jumpTo: 28
20 <89> pushThisContext:
21 <75> pushConstant: 0
```

```
22 <C8> send: blockCopy:
23 <A4 02> jumpTo: 27
25 <10> pushTemp: 0
26 <7D> blockReturn
27 <7D> blockReturn
28 <69> popIntoTemp: 1
29 <76> pushConstant: 1
30 <68> popIntoTemp: 0
31 <11> pushTemp: 1
32 <C9> send: value
33 <69> popIntoTemp: 1
34 <77> pushConstant: 2
35 <68> popIntoTemp: 0
36 <11> pushTemp: 1
37 <C9> send: value
38 <7C> returnTop
```

## Exercise 11.5:

```
| irMethod aCompiledMethod res |
irMethod:= IRBuilder new
    numRargs:  1;
    addTemps: #(self i);      "receiver and args"
    pushLiteral: 0;
    storeTemp: #i;
    popTop;
    jumpBackTarget: #back;
    pushTemp: #i;
    pushLiteral: 10;
    send: #<;
    jumpAheadTo: #continue if: false;
    pushLiteral: Transcript;
    pushTemp: #i;
    pushLiteral: 1;
    send: #+;
    storeTemp: #i;
    send: #asString;
    send: #show:;
    popTop;
    jumpBackTo: #back;
    jumpAheadTarget: #continue;
    returnTop;
    ir.

aCompiledMethod  := irMethod compiledMethod.
```

```
aCompiledMethod valueWithReceiver: 3@4 arguments: #().
```

### Exercise 11.6:

```
InstructionClient subclass: #SendCounter
    instanceVariableNames: 'scanner count'
    classVariableNames: ''
    poolDictionaries: ''
    category: 'Bytecode'

SendCounter class >> on: aMethod
    ^ self new on: aMethod

SendCounter >> initialize
    count := 0

SendCounter >> on: aMethod
    "Append to the stream, aStream, a description of each bytecode in the
    instruction stream."

    | end |
    scanner _ InstructionStream on: aMethod.
    end _ aMethod endPC.
    [scanner pc <= end]
        whileTrue: [scanner interpretNextInstructionFor: self].
    ^count.

SendCounter >> send: selector super: supered numArgs: numberArguments
    count := count + 1.

self assert: (SendCounter on: (Object compiledMethodAt: #halt)) = 1.
```

### Exercise 11.7:

```
ContextPart class >> numberOfBytecodeExecuted: aBlock
    | tallies |
    tallies := 0.
    thisContext sender
        runSimulated: aBlock
        contextAtEachStep: [:current | tallies := tallies + 1].
    ^tallies

self assert:
    (ContextPart numberOfBytecodeExecuted: [3.14159 printString]) =  1039.
```

### Exercise 11.8:

```
ContextPart class >> bytecodeCovered: aBlock
    | tempDict result cm method percentage |
    tempDict := Dictionary new.
    thisContext sender
        runSimulated: aBlock
        contextAtEachStep:
            [:cur | | tmpMethod |
                cur selector ~= #DoIt ifTrue: [
                    tmpMethod := cur methodClass >> cur selector.
                    (tempDict includesKey: tmpMethod) ifFalse:
                        [tempDict at: tmpMethod put: Set new].
                (tempDict at: tmpMethod) add: cur pc]].

    result := OrderedCollection new.
    tempDict associationsDo: [:assoc | | total |
        cm := assoc key.
        total := cm endPC - cm initialPC + 1.
        total ~~ 0 ifTrue: [
            percentage := (assoc value size * 100 / total) asInteger.
            percentage > 100 ifTrue: [self halt].
            method := assoc key.
          result add:  method methodClass name, '>>', method selector.
          percentage]].
    ^ result asArray

| res |
res := ContextPart bytecodeCovered: [3.14159 printString].
self assert: res size = 34.
```