2. Exemplary Solutions: Objects and Expressions

Exercise 2.1: Simple Expressions

Table 1: Solution exercise 2.1

<table>
<thead>
<tr>
<th>Expression</th>
<th>Receiver</th>
<th>Selector</th>
<th>Arguments</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 + 4</td>
<td>3</td>
<td>+</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Date today</td>
<td>Date (class!)</td>
<td>today</td>
<td>None.</td>
<td>(current date)</td>
</tr>
<tr>
<td>anArray at: 1</td>
<td>anArray</td>
<td>at:put:</td>
<td>1 and 'hello'</td>
<td>an Array with 'hello' as first element</td>
</tr>
<tr>
<td>put: 'hello'</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25@50</td>
<td>25</td>
<td>@</td>
<td>50</td>
<td>a Point: 25@50</td>
</tr>
</tbody>
</table>

Exercise 2.2: Some Questions

- Objects described by the following expressions are:
  1. ‘Hello, Dave’
     is a String
  2. #Node1
     is a Symbol
  3. #(1 2 3)
     is an Array with 1, 2, and 3 as elements

- The following code:

```smalltalk
| anArray |
anArray := #('first' 'second' 'third' 'fourth').
ˆanArray at: 2
```

yields the String ‘second’ when evaluated.

Exercise 2.3:

- Minimal number of parentheses for the following expressions:
  1. 3 + 4 + (2 * 2) + (2 * 3)
  2. x isZero ifTrue: [...].
     (x includes: y) ifTrue: [...].

- Results of the following expressions

```
6 + 4 / 2 = 5
1 + 3 negated = -2
1 + (3 negated) = -2
2 raisedTo: 3 + 2 = 32
2 negated raisedTo: 3 + 2 = -32
```
Exercise 2.4:

- Sequence of executions steps for the following expressions:

  1. Date today daysInMonth
     (a) sending message `today` to class `Date`, resulting in the current date.
     (b) sending message `daysInMonth` to this current date object, resulting in the number of
days in this month (eg. 30 for September).

  2. `#(1 2 3)` size + 7
     (a) creating an array with elements 1, 2 and 3. Internally, the Smalltalk compiler translates
     the expression `#(1 2 3)` to `Array with: 1 with: 2 with: 3`
     (b) sending message `size` to this array object, resulting in the SmallInteger 3.
     (c) sending message `+` with argument 7 to 3, resulting in the SmallInteger 10.

  3. `5@5 extent: 6.0 truncated @ 7`
     (a) sending message `@` to 5 with argument 5, resulting in the point `5@5`.
     (b) sending message `extent:` to this point. But now Smalltalk will first evaluate the
     argument expression passed to `extent:`.
     (c) sending message `truncated` to 6.0 (a float), resulting in the SmallInteger 6.
     (d) sending message `@` to 6 with argument 7, resulting in the point `6@7`.
     (e) Now the argument for `extent:` has been completely evaluated, thus Smalltalk sends
     the message `extent:` to point `5@5` with argument point `6@7`, resulting in a rectangle
     with origin `5@5` and corner `11@12`.

- Transcript show: `34 + 89 printString`
  prints the sum of `34 + 89` (that is, 123) on the Transcript.