Parallel Programming WS 2014/2015 - Assignment 7

Kastens, Pfahler
Institut für Informatik, Fakultät für Elektrotechnik, Informatik und Mathematik, Universität Paderborn
Feb. 02, 2015

Exercise 1 (Synchronous Message Passing)

Consider the following CSP program:

```
process Compare
   var i : integer;
       x : (1..10)integer;
   i := 1;
   do i <= 10 -> if x(i) >= 0 -> skip;
                 [] x(i) < 0  -> Count!x(i);
                 fi
                 i := i + 1;
   od
end

process Count
   var y, sum : integer;
   sum := 0;
   do Compare?y -> sum := sum + y;
   od;
   Print!sum
end
```

What does this program compute? How does it terminate?

Exercise 2 (Synchronous Message Passing)

Write a system of processes "Even", "Odd", and "Sum" in the above notation.

- "Even" sends the integer numbers 2, 4, .. 100 to process "Sum"
- "Odd" sends the integer numbers 1, 3 .. 99 to process "Sum"
- "Sum" adds all incoming numbers and sends the final result to standard process "Print".

Exercise 3 (Distributed Systems: Probe and echo)

Directory `blatt7/ProbeNet` contains a framework for probe and echo operations in a net (see Slide 73, Slide 74). Apply the probe and echo approach to compute the sum of numbers where one number is stored locally at each node of a net.

The framework consists of the following important classes:

- `Node.java`: a node of the net with associated input port (channel); stores one operand of the sum
- `Message.java`: represents the three kinds of messages needed (probes, dummies, echoes)
- `Main.java`: builds an example net, creates and starts a `Prober` thread for each node

Complete the `run()` method of the inner class `Main.Prober` according to the algorithm outlined on Slide 74. The code for the initiator node is provided. Test your implementation using different initiators.