SOFTWARE ENGINEERING

RWTH AACHEN UNIVERSITY

http://www.se.rwth-aachen.de/

Department of Software Engineering
RWTH Aachen University
Prof. Dr.-Ing. Manfred Nagl, Emeritus,
Dipl.-Math. Michael von Wenckstern
(vonwenckstern@se-rwth.de)

Exercise course: *Ada*
WS 2014 / 15
October 10, 2014

Exercise Sheet 1

Submission:

When:	Monday, **October 20, 2014**. 11:59 pm
Where:	L^{2}P-eLearning room of Ada or e-mail

Organization

Exercise sheets must be submitted in groups of two to four students. The submission must be delivered electronically via the L^{2}P-eLearning room of Ada.

Exercise 1.1 Development Environment

During the exercise courses there will be programming tasks. Therefore, get familiar with a development environment for Ada. Install an Ada compiler and a graphical IDE supporting Ada.

The standard compiler for Ada is GNAT (Gnu Ada Translator) which is available for different platforms. GNAT Programming Studio is a recommendable graphical IDE for Windows. For Linux the Emacs-based GLIDE is recommendable and already included in the GNAT distribution. In http://en.wikibooks.org/wiki/Ada_Programming/Installing you will find download links, instructions for installation, and some helpful information in programming with Ada. For Mac OS X there are on http://www.macada.org/macada/Welcome.html plugins for Apples IDE "XCode".

GNATbench, an Ada Plug-in for Eclipse, can be downloaded with the GNAT GPL Edition at http://libre.adacore.com/download/. It requires Eclipse Platform Version 3.6 - 3.7 or 4.2.

Exercise 1.2 Programming Task (1 Point)

Write a „Hello World!"-program to test your development environment.
Commit the source code and a screenshot of your running IDE.

Exercise 1.3 Derivation Tree (6 Points)

Get familiar with the Ada EBNF and write down the derivation tree according to the Ada EBNF for the following package.

```
WITH Ada.Text_IO;

PACKAGE BODY Mobile IS

  FUNCTION transmit (

        msg : IN  string)

    RETURN integer IS


  BEGIN

    Ada.Text_IO.Put_Line("Sending Message ...");

    RETURN 0;

  END transmit;


END Mobile;
```

For that use the **XML-notation**. In the following, the beginning of the solution is presented. Here, you can detect the desired structure. All nonterminal symbols are XML-tags, all terminal symbols are contents of the tags. You can find the syntax in Appendix 6 of the Ada-book or at http://www.seas.gwu.edu/~adagroup/ada95-syntax/ , http://www.dwheeler.com/lovelace/s2s4.htm

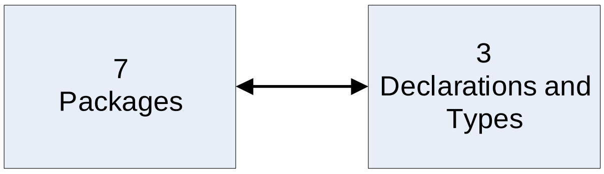
```
<compilation_unit>
  <context_clause>
    <with_clause>
      WITH
      <library_unit_name>Ada.Text_IO</library_unit_name>
      ;
    </with_clause>
  </context_clause>
  <library_item>

        ...

  </library_item>
</compilation_unit>
```

To save work you can end with the parsing if the nonterminal symbol ends with **name**, **identifier** or **expression** (e.g., for **<library_unit_name>**).

It is suggested to this work with an XML-Editor and copy the solution later on into the text document in order to avoid spelling errors.

Exercise 1.4 Relations in the Ada EBNF (6 Points)

The Ada EBNF is divided into sections describing syntactical constructs which belong together. Inside one section there are numerous relations between such constructs. But there are also relations between different sections. Draw a diagram for sections 2 to 13, similar to the following one, containing all direct relations derived from the syntax rules.



An arrow goes from section 3: *Declarations and Types* to section 7: *Packages*, because rule 3.11 references the nonterminal **package_body** of rule 7.2:

```
3.11: proper_body ::=
         subprogram_body
       | package_body                       ← from 7.2: Package Bodies
       | task_body
       | protected_body
```

Vice versa an arrow goes from section 7 to section 3, because rule 7.1 *Package Specifications and Declarations* refers **basic_daclarative_item** of rule 3.11:

```
7.2: package_specification ::=
       PACKAGE defining_program_unit_name IS
       {basic_declarative_item}             ← from 3.11: Declarative Parts
   [private {basic_declarative_item}]
   END [[parent_unit_name.]identifier]
```