Safe and Secure Software
Welcome to the Problem Sessions!

WS2016/17

Prof. Stefan Lucks, Eik List

`<firstname>.<lastname>(at)uni-weimar.de`

Bauhaus-Universität Weimar

October 10, 2016

Welcome!

Welcome to the problem session of **Safe and Secure Software.**

What will it be about?

- Learning Ada basics and later intermediate concepts

Welcome!

Welcome to the problem session of **Safe and Secure Software.**

What will it be about?

- Learning Ada basics and later intermediate concepts
- Formally proving software correctness

Welcome!

Welcome to the problem session of **Safe and Secure Software.**

What will it be about?

- Learning Ada basics and later intermediate concepts
- Formally proving software correctness
- Testing your software

Welcome!

Welcome to the problem session of **Safe and Secure Software.**

What will it be about?

- Learning Ada basics and later intermediate concepts
- Formally proving software correctness
- Testing your software

Welcome!

Welcome to the problem session of **Safe and Secure Software.**

What will it be about?

- Learning Ada basics and later intermediate concepts
- Formally proving software correctness
- Testing your software

Actually the lecture where you can learn proper testing

First Assignment

- About getting familiar with Ada
- Installing the GNAT toolkit
- Not graded

Problem Sets

- You will hardly learn Ada without working on the problem sets
- Learning groups of two (or three) persons are encouraged
- From the 2nd problem we will offer mini-projects
- **Presentations:**
  - Everyone has to solve **(at least) two** and **present one** mini-project (15-20 minutes)
- **Submission Deadline for Problem Sets:**
  - Saturday before the problem session by e-mail 12:00 (noon).
- The new problem set will usually be published Tu/We after the problem session

Apply for a Mini Project

Deadline:	Friday before the problem session, 12:00 (noon).
Policy:	First come first serve.
Notification:	Via email and on the website of the problem session.

Final Grade Bonus

- Achieve $\geq$ 25% of the points for each problem set
  $\implies$ 1/3 grade bonus
- After the end of lectures, bonus projects will be available
- Solve them and get an additional 1/3 grade bonus.

Comments on Source Code – Intendation

Works, but not readable.

```
          with Ada.Text_IO;

          procedure Add_And_Hello is Left, Right, Result: Integer := 0;
          begin Result := Left + Right; Ada.Text_IO.Put(Result'Img);
          Ada.Text_IO.Put_Line("Hello World"); end Add_And_Hello;
```

Comments on Source Code – Intendation

Works, but not readable.

```
           with Ada.Text_IO;

           procedure Add_And_Hello is Left, Right, Result: Integer := 0;
           begin Result := Left + Right; Ada.Text_IO.Put(Result'Img);
           Ada.Text_IO.Put_Line("Hello World"); end Add_And_Hello;
```

Better:

```
with Ada.Text_IO;

procedure Add_And_Hello is
  Left, Right, Result: Integer := 0;
begin
  Result := Left + Right;
  Ada.Text_IO.Put(Result'Img);
  Ada.Text_IO.Put_Line("Hello World");
end Add_And_Hello;
```

Comments on Source Code – Naming

You can only guess what this packages is supposed to do.

```ada
package Stuff is
  type Element is record
    X, Y, Z: Float := 0.0;
  end record;

  function Check1(Left: Element; Right: Element) return Boolean;
  function Check2(Left: Element; Right: Element) return Boolean;
  function Compute(Left: Element; Right: Element) return Element;
  function FromTo(Item: Element) return Float;
end Stuff;
```

Comments on Source Code – Naming

You can only guess what this packages is supposed to do.

```ada
package Stuff is
  type Element is record
    X, Y, Z: Float := 0.0;
  end record;

  function Check1(Left: Element; Right: Element) return Boolean;
  function Check2(Left: Element; Right: Element) return Boolean;
  function Compute(Left: Element; Right: Element) return Element;
  function FromTo(Item: Element) return Float;
end Stuff;
```

Better...Ah...it's about vectors.

```ada
package Vectors is
  type Vector is record
    X, Y, Z: Float := 0.0;
  end record;

  function Are_Equal(Left: Vector; Right: Vector) return Boolean;
  function Are_Orthogonal(Left: Vector; Right: Vector) return Boolean;
  function Cross_Product(Left: Vector; Right: Vector) return Vector;
  function Distance_To_Origin(Item: Vector) return Float;
end Vectors;
```

Comments on Source Code – Use Clause

Source code can become unclear with global use clauses.

```
with Ada.Text_IO, Ada.Integer_Text_IO, Ada.Float_Text_IO;
use Ada.Text_IO, Ada.Integer_Text_IO, Ada.Float_Text_IO;

procedure Calculator is
  Left, Right, Result_A: Integer := 0;
  Result_Q: Float := 0.0;
begin
  Result_A := Left + Right;
  Result_Q := Float(Left)/Float(Right);

  Put(Result_Q + Result_Q); -- Ada.Float_Text_IO
  Put(Result_A + Result_A); -- Ada.Integer_Text_IO
  Put(Result_A'Img);        -- Ada.Text_IO
end Caculator;
```

Comments on Source Code – Use Clause

The local the better. Or even more better: package renaming.

```ada
with Ada.Text_IO, Ada.Integer_Text_IO, Ada.Float_Text_IO;

procedure Calculator is
  package ATIO renames Ada.Text_IO;
  package AIIO renames Ada.Integer_Text_IO;
  package AFIO renames Ada.Float_Text_IO;

  use ATIO; -- local use clause

  Left, Right, Result_A: Integer := 0;
  Result_Q: Float := 0.0;

begin
  Result_A := Left + Right;
  Result_Q := Float(Left)/Float(Right);

  AFIO.Put(Result_Q + Result_Q); -- Ada.Float_Text_IO
  AIIO.Put(Result_A + Result_A); -- Ada.Integer_Text_IO
  Put(Result_A'Img);             -- Ada.Text_IO (ATIO)
end Caculator;
```

Submission Guidelines

- Follow the **Ada Style Guide**:
  `http://en.wikibooks.org/wiki/Ada_Style_Guide`

Submission Guidelines

- Follow the **Ada Style Guide**:
  `http://en.wikibooks.org/wiki/Ada_Style_Guide`
- Submit your source code as **.adb, .ads** (not as .txt, .pdf).

Submission Guidelines

- Follow the **Ada Style Guide**:
  `http://en.wikibooks.org/wiki/Ada_Style_Guide`
- Submit your source code as **.adb, .ads** (not as .txt, .pdf).
- If you use an IDE, you can also submit a .gpr (project) file

Submission Guidelines

- Follow the **Ada Style Guide**: `http://en.wikibooks.org/wiki/Ada_Style_Guide`
- Submit your source code as **.adb, .ads** (not as .txt, .pdf).
- If you use an IDE, you can also submit a .gpr (project) file
- **Do not submit executables and temporary files** (.ali, .o, .exe,. . .)

Submission Guidelines

- Follow the **Ada Style Guide**:
  `http://en.wikibooks.org/wiki/Ada_Style_Guide`
- Submit your source code as **.adb, .ads** (not as .txt, .pdf).
- If you use an IDE, you can also submit a .gpr (project) file
- **Do not submit executables and temporary files**
  (.ali, .o, .exe,. . .)
- If you submit more than one file, put them in an archive (tar.gz or .zip) and name it as follows:
  `<name>-<matrnumber>.tar.gz|.zip`
  (one name per group suffices)

Submission Guidelines

- Follow the **Ada Style Guide**:
  `http://en.wikibooks.org/wiki/Ada_Style_Guide`
- Submit your source code as **.adb, .ads** (not as .txt, .pdf).
- If you use an IDE, you can also submit a .gpr (project) file
- **Do not submit executables and temporary files** (.ali, .o, .exe,. . .)
- If you submit more than one file, put them in an archive (tar.gz or .zip) and name it as follows:
  `<name>-<matrnumber>.tar.gz|.zip`
  (one name per group suffices)
- All group members have to be mentioned in the documentation (if any) or in the source file

Submission Compiler Options

- Use the following compiler switches to ensure proper coding/testing:
  - `-gnatwa`: Shows all warnings
  - `-gnatwe`: Threats all warnings as errors
  - `-gnato`: Overflow checking
  - `-gnat95|05|12`: Specifies the Ada version (Ada95, Ada05, or Ada12)

Submission Compiler Options

- Use the following compiler switches to ensure proper coding/testing:
  - `-gnatwa`: Shows all warnings
  - `-gnatwe`: Threats all warnings as errors
  - `-gnato`: Overflow checking
  - `-gnat95|05|12`: Specifies the Ada version (Ada95, Ada05, or Ada12)
- **Submit before the deadline!**

Questions?

https://frinkiac.com