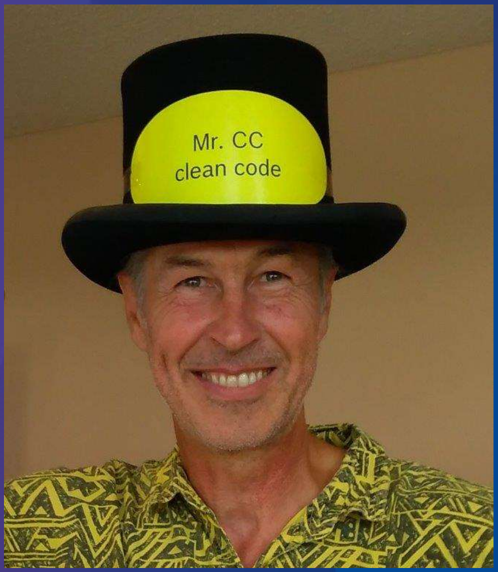
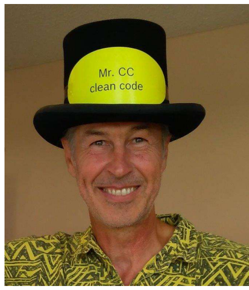
C++ BASICS LIFECYCLE SCOPE & VISIBILITY



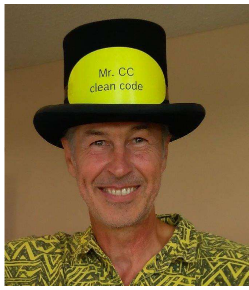
Mr. CC clean code
Gerd Hirsch (CC-AD/ESW1)

C++ Basics: Lifecycle
Prerequisites



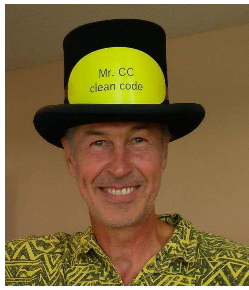
- This is a hands-on workshop, you need to have
  - an installed and running development environment
  - an editor ready to edit source code
  - the ability to compile and execute your program
  - basic knowledge of the C++ syntax

C++ Basics: Lifecycle
Prerequisites



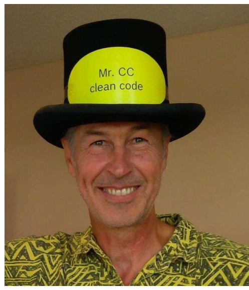
- This Workshop
  - focuses on the mechanisms of C++ in the context of the life cycle of objects
  - and touches briefly on related topics
  - is structured in **Sessions** that build on each other
- Not addressed by this workshop
  - How to find the appropriate classes representing the right abstractions
    is part of **Object oriented Analysis and Design with the UML (OOAD)**
  - How to implement these classes to get rid of the dependencies
    is part of **Object oriented Design Principles and Patterns with the UML (OODP)**

C++ Basics: Lifecycle
Prerequisites



- Create for each Session a separate project Session<1..n> and a complete set of sourcefiles
  - Do nothing else but what is adviced in the exercises
  - we want to see the compiler messages
  - Use the valuable examples for further experiments in your daily work
- Each Session will have
  - a theory part with exercises and
  - a separate Solutions part
  - **Do first the exercises before** you take a look at the solution part
- References
  - Some links to online resources in the sessions
  - In German: chapter numbers of the C++ Script OO_CPP_Schulung.pdf

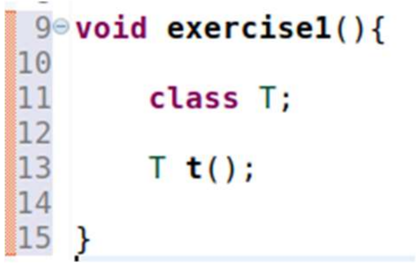
C++ Basics: Lifecycle
Prerequisites



- ...and please let me know quickly whether it works for you how we do it
  - **Not just at the end of all sessions**

- the compressed sources are due to the limited space on the slides
  - Try to write beautiful code!
  - Source code is formated like this:
    void function(int i){ /*functionbody*/}

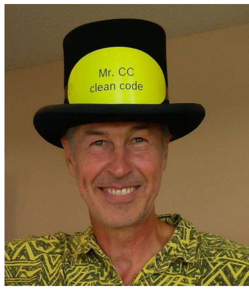
    or like this:



```
void exercise1(){

    class T;

    T t();

}
```

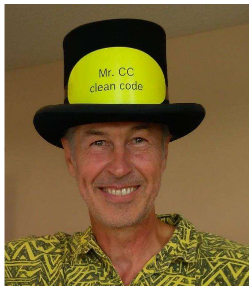
BOSCH

C++ Basics: Lifecycle
Agenda



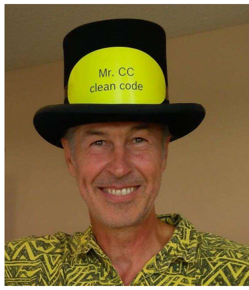
- Session 1
  - Tools and Activities, simplified Process Model
  - Exercises: Warm up; Some Experiments with Compilers and Translation Units
  - Solutions
- Session 2
  - Compiler synthesized Methods and Rules
  - Exercises and Solutions
- Session 3
  - Customizing compiler synthesized Method and their Signatures
  - L-Value qualified assignment operator
  - Exercises and Solutions

C++ Basics: Lifecycle
Agenda



- Session 4
  - Copy Elision and the (Named) Return Value Optimization (N)RVO
  - Function call parameter / return types and dangling references
- Session 5
  - Compiler synthesized Methods with Member/Attributes and Base Classes
- Session 6
  - Customizing Compiler synthesized Methods with Member/Attributes and Base Classes
  - The rule of Six
- Session 6a
  - a brief overview of dynamic polymorphism, references and pointers
  - Type Slicing

C++ Basics: Lifecycle
Agenda



- Session 7
  - User defined Conversions
    constructors, cast operators and the Keyword **explicit**
- Session 8
  - a brief Introduction to handling Resources and RAII
  - Implementing a ResourceHandler
- Session 9
  - a brief Introduction to Templates
  - UniquePointer, a first Template
  - templatized Conversion Constructors and Assignment Operators