Prof. Dr. Jasmin Blanchette
Dr. Martin Desharnais-Schäfer
Dr. Michael Kirsten
Elisabeth Lempa

Ludwig-Maximilians-Universität München
Institut für Informatik
Discussion on 26.11.2025
Homework due on 03.12.2025 at 16:00

Possible solution for Exercise Sheet 7 in

Scientific and Technical English for Computer Scientists

The exercise sheets consist of in-class exercises and homework. The in-class exercises take place during the second half of the lecture time slots. The homework, which is optional and ungraded, can be submitted via the "Homework" section in Moodle. The homework is subject to peer review.

Unless indicated otherwise, generative artificial intelligence assistants such as ChatGPT may be used, as long as you acknowledge how you use them as specified by the Institute's policy on plagiarism.[1] However, you may not use such tools to generate peer reviews for you. In addition, we strongly recommend that you do not use them to generate entire solutions, since that would defeat the purpose of the exercises.

Homework 7-3 ***Commas Galore*** Write a text arguing the merits of either static type systems (as found in, e.g., C++ and Java) or dynamic type systems (as found in, e.g., JavaScript and Python) using as many different types of commas as possible. Use them correctly. Your text should be at most 250 words long.

POSSIBLE SOLUTION:

All software developers who value software correctness and reliability,$_{\text{vocative}}$ please consider using a statically typed programming language for your next project.

Static type systems,$_{\text{bracketing}}$ as found in languages such as Go,$_{\text{listing}}$ Haskell,$_{\text{listing}}$ Java,$_{\text{serial}}$ and Rust,$_{\text{bracketing}}$ ensure at compile-time that programs manipulate values in a meaningful way. As Benjamin C. Pierce wrote,$_{\text{separating}}$ "A type system is a tractable syntactic method for proving the absence of certain program behaviors by classifying phrases according to the kind of values they compute." By enforcing type constraints,$_{\text{introductory}}$ these systems transform potential runtime errors into compile-time errors,$_{\text{joining}}$ and entire classes of bugs are ruled out before any code execution.

Another use of type systems is as verified documentation. For example,$_{\text{introductory}}$ annotating a variable as immutable or a function as pure not only provides some useful information to readers,$_{\text{bracketing}}$ which might be your future self,$_{\text{bracketing}}$ but

[1] `https://www.medien.ifi.lmu.de/lehre/Plagiate-IfI.pdf`

also tells the development tools to check and enforce these properties.

Some static type systems are considered too restricting or verbose,joining but modern ones have evolved beyond rigid constraints and incorporate powerful features such as type inference,listing parametric polymorphism,serial and algebraic data types.

In essence,introductory static typing is not a limitation; it is a safeguard that protects users from errors,listing offers helpful guidance to developers,serial and enables thinking about complex problems in terms of abstractions.