Prof. Dr. Annette Bieniusa
Philipp Lersch, M. Sc.

**RPTU Kaiserslautern-Landau**
**Fachbereich Informatik**
**AG Softwaretechnik**

# Exercise 3: Programming Distributed Systems (Summer 2025)

**Submission** Deadline: 20.05.2025 AOE

- In your Git repository, create a branch for this exercise sheet (for example with `git checkout -b ex3`)
- Create a folder named "ex3" in your repository and add your solutions to this folder.
- Create a merge request in Gitlab and assign Philipp Lersch as assignee. If you do not want to get feedback on your solution, you can merge it by yourself right away. Otherwise, you can merge if after feedback was given.

## 1 Specifying Causal Order

Alice and Bob come up with two different variants of the causal-order property:

**Causal-Broadcast** $CB_A$ If $p_i$ delivers $m$, then $p_i$ must first deliver every message $m'$ with $m' \rightarrow m$.

**Causal-Broadcast** $CB_B$ If $p_i$ delivers $m'$ and $m$ and $m' \rightarrow m$, then $p_i$ must deliver $m'$ before $m$.

- Give an examplary execution to show that $CB_A$ and $CB_B$ are not equivalent.
- Which one is more general, i.e. does $CB_A \Rightarrow CB_B$ or $CB_B \Rightarrow CB_A$?

## 2 Code snippets

Look into the module *Snippets* and answer the given questions by adding new comments.
Feel free to open up a shell and play with the code.
Save your progress before executing $q3setup/0$.
This snippet might take some time even if it seems to be done.

# 3 Simple Vectorclock Server

Your task is to implement a `VCServer` module as a server process step by step. In this task, we will use Elixir's `receive` operator and `send` and `spawn` functions.

Check the exercise slides for additional guidance.

1. Within the VCServer module: Implement the $start\_link/0$ function to spawn a new process which executes the $loop/1$ function.
2. Within the VCServer module: Extend the $loop/1$ function with an receive statement that listens for a message containing : $ping$ and prints "pong!". Do not forget to assign the result of the receive statement with "$new\_state = receive...$" as we will need it later.
3. Within the VCServer module: Extend the $loop/1$ function such that multiple :ping messages can be received
4. Within $iex -S mix$: Create the server process through $start\_link$ and send a few : $ping$ messages.
5. Within the VCServer module: Extend the $loop/1$ function such that it also matches for $\{sender, : new\}$, $\{sender, : increment\}$, $\{sender, : get\}$, $\{sender, : leq, other\_vc\}$, $\{sender, merge, other\_vc\}$. These should all make use of the provided $Vectorclock$ module. Additionally the result of the operation should be send to the original sender and saved as new state of the process. In this example the state is a vectorclock
6. Within $iex -S mix$: Try your solution interactively
7. Within the VcserverTest module: Write three or more tests for your implementation
8. Add type specification

# 4 GenServer Vectorclock

Now we will implement the server process from the previous task as $VCGenServer$.

1. Take a look at the offical documentation at `https://hexdocs.pm/elixir/GenServer.html`
2. Within the VCGenServer module: The given callbacks are just an example to show the required structure and can be freely modified and deleted.
3. Within the VCGenClient module: The given execution is an example on how to interact with an GenServer implementation. It can also be freely modified and deleted. It is provided to help you understand how to interact with a GenServer implementation. You can use it as scratch board while development to save some time by not rewriting client logic within the console.
4. Setup the init callback such that the VCGenServer's state is initialized with the passed-as-argument vectorclock or Vectorclock.new()
5. Introduce $handle\_call$ and $handle\_cast$ callbacks for the following messages : $new$, : $increment$, : $get$, $\{: get, pid\}$, $\{: leq, other\_vc\}$, $\{: merge, other\_vc\}$. Use the provided Vectorclock module. Each callback must return the new state as vectorclock to the GenServer.
6. Add type specification
7. Add three or more tests