Assignment 9.1 Quiz

1. Let $C$ be a class, composed from the Mixins $M$ and $N$. Suppose, $M$ and $N$ both implement a method $f()$. Is it true that

- The conflicting methods $f()$ from $M$ and $N$ lead to a compiler error and have to be resolved manually
- There is no compiler error, but one implementation of $f()$ from $M$ or $N$ overwrites the other

2. Now assume, that $M$ and $N$ are Traits instead of Mixins. Is it true that

- The conflicting methods $f()$ from $M$ and $N$ lead to a compiler error and have to be resolved manually
- There is no compiler error, but one implementation of $f()$ from $M$ or $N$ overwrites the other

3. Given the traits $T_1, T_2$ and class $C$:

$$T_1 = \{ a \mapsto 1 \}$$
$$T_2 = \{ a \mapsto 2 \}$$
$$C = \langle \emptyset, \text{nil} \triangleright (T_1 + T_2) \rangle$$

What is the value of $(C \triangleright T_1)(a)$?

- $\top$
- $1$
- $\perp$

4. (Attention: Several answers might be true for this question!)

$c_1 \sqcup c_2 = c_1 \sqcup c_2$ is true for

- $c_1 = \{ a = 0x1 \}, \ c_2 = \{ b = 0x1 \}$
- $c_1 = \text{mixin}(c_3)(c_2), \ c_2 = \{ a = 0x1 \}, \ c_3 = \{ a = 0x2 \}$
- $c_1 = \text{mixin}(c_2)(c_3), \ c_2 = \{ a = 0x1 \}, \ c_3 = \{ a = 0x2 \}$
- $c_1 = \text{mixin}(c_3)(c_4), \ c_2 = c_3 \triangleright c_4, \ c_3 = \{ a = 0x1 \}, \ c_4 = \{ a = 0x2 \}$

5. Why is exclusion an important composition operator for Traits?
Assignment 9.2 Having fun with Mixins
Reconsider the example from the lecture about synchronized file- and socket-streams. The following classes are given:

\[
\begin{align*}
\text{FileStream} &= \{ \text{read} = 0x1, \text{write} = 0x2 \} \\
\text{SocketStream} &= \{ \text{read} = 0x3, \text{write} = 0x4 \} \\
\text{SyncRW} &= \{ \text{read} = 0x5, \text{write} = 0x6 \}
\end{align*}
\]

Your task is to come up with a new class \textit{SynchedFileStream} which mixes the class \textit{SyncRW} into the class \textit{FileStream}.

Assignment 9.3 Mixins Ruby
Implement the \textit{Stream Wrapper} scenario from the lecture based on Ruby Mixins

Assignment 9.4 Implementation differences: Traits vs. Mixins
A next mainstream implementation of traits comes with the virtual extension methods in Java 8.

- Implement a solution for the \textit{Stream Wrapper} problem. You may use the following code:

```java
interface Stream {
    int read();
}

interface FileStream extends Stream {
    default int read() { /* ... */ }
}

interface NetworkStream extends Stream {
    default int read() { /* ... */ }
}

interface Synch {
    default void acquireLock() { /* ... */ }
    default void releaseLock() { /* ... */ }
}
```

- Compare your solution to the one based on Mixins from the above assignment. What are the differences? Which one is more flexible w.r.t. software engineering aspects?