9.1 The inner-statement

Description

Inner also works for classes, so we have to include that some way! Perhaps we may just put a note here saying that inner works in the same way for classes.

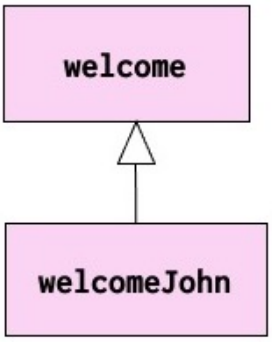
First we introduce the `inner`-statement, which is central for defining methods that may be used as supermethods of other methods. We start by a simple example of a method `welcome` that uses `inner`:

```
welcome:
   console.print("Welcome: ")
   inner(welcome)
   console.print("Have a nice day!")
```

Execution of `welcome` implies that `console.print("Welcome: ")` is executed, followed by the execution of `inner(welcome)`. If `welcome` is executed by an invocation `welcome`, then execution of `inner(welcome)` is an action that has no effect.

Let us next consider the situation where `welcome` is a supermethod of some method like `welcomeJohn` below:

```
welcomeJohn: welcome
   console.print("John. ")
```

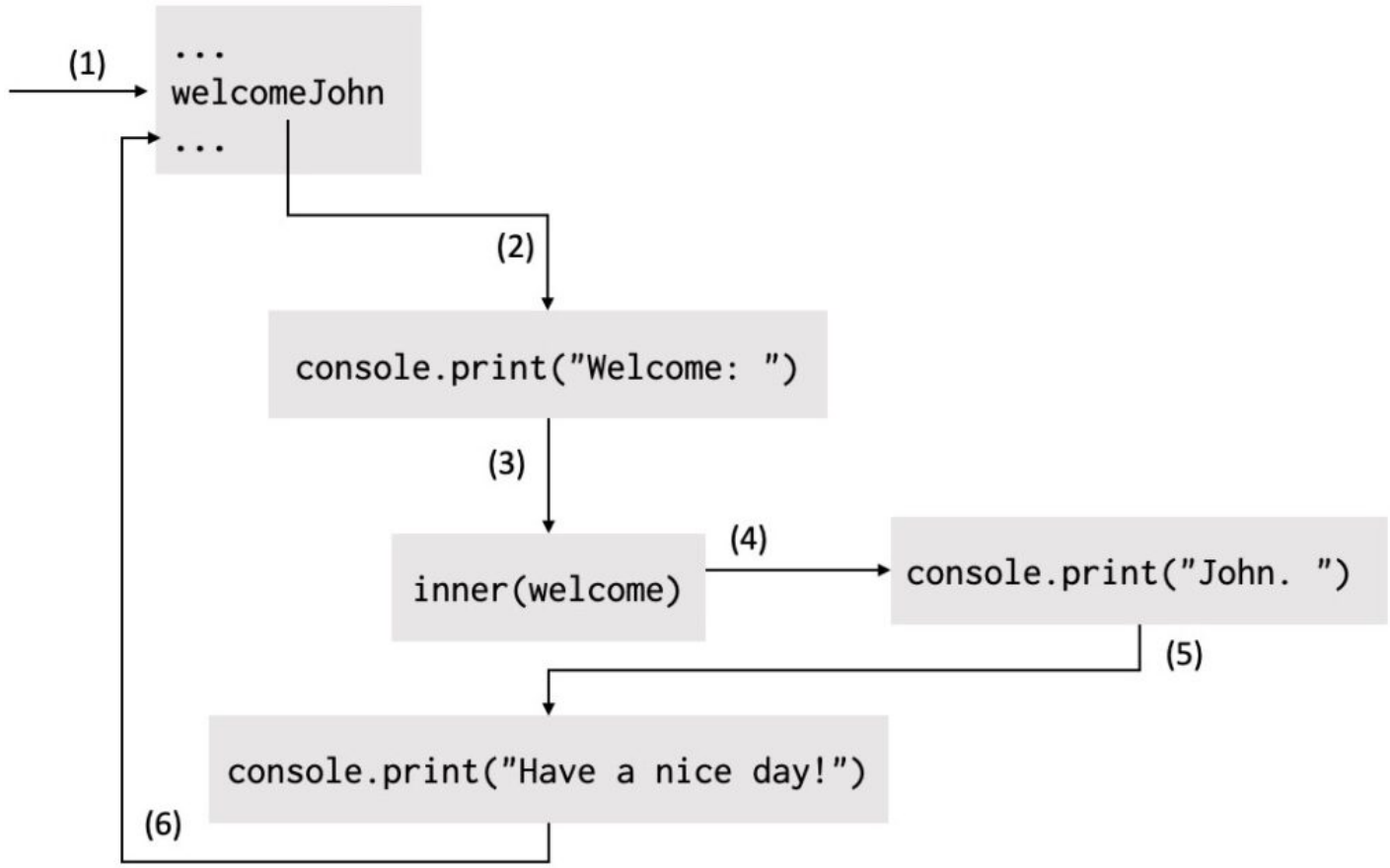


welcome as a supermethod to welcomeJohn

An invocation of `welcomeJohn` implies that execution starts with the statements of the supermethod, here `welcome`. Execution of `inner(welcome)`, then implies that the statements in the main part of `welcomeJohn` are executed. In this case there is only one statement, `console.print("John. ").`

```
...
welcomeJohn...
```

After execution of this statement, control returns to after `inner` in `welcome` and `console.print("Have a nice day!")` is executed.



Sequence of actions for the call of the method “welcomeJohn”

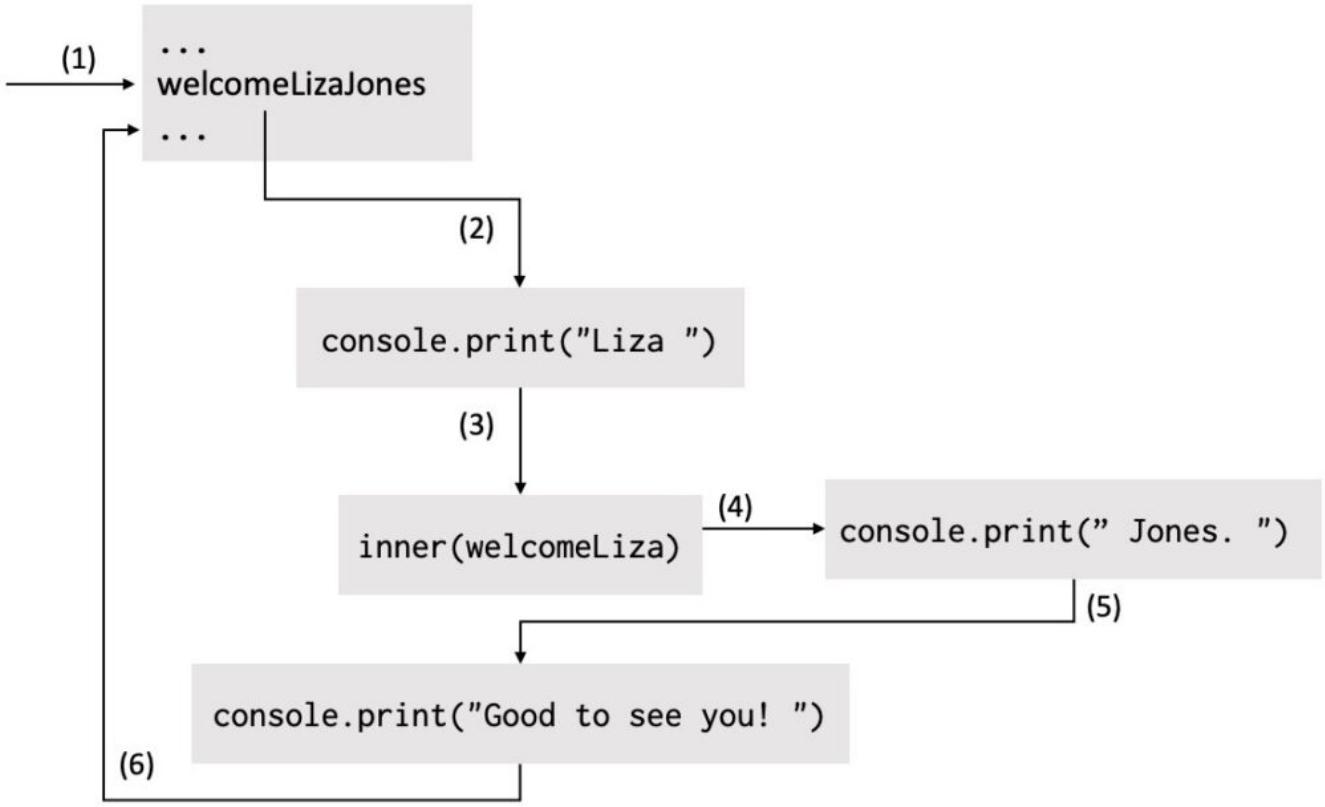
We may define other submethods of `welcome` as:

```
welcomeLiza: welcome
   console.print("Liza ")
   inner(welcomeLiza)
   console.print("Good to see you! ")
```

As can be seen, `welcomeLiza` includes an `inner(welcomeLiza)`. We may a submethod of `welcomeLiza`:

```
welcomeLizaJones: welcomeLiza
   console.print(" Jones. ")
```

Execution of `welcomeLizaJones` then takes place as shown in the figure below:



Sequence of actions for the call of the method “welcomeLizaJones”