Programming (ERIM)

Lecture 6: Programming by contract

Tommi Tervonen

Econometric Institute, Erasmus School of Economics

Programming by contract

- Methods define a *contract* between the supplier (you) and the consumer (you or someone else)
- Contract **partially** defined through the signature:

```
function arr = sortArrayFromIndex(array, index)
```

Programming by contract

- Methods define a *contract* between the supplier (you) and the consumer (you or someone else)
- Contract **partially** defined through the signature:

```
function arr = sortArrayFromIndex(array, index)
```

Contract:

1. The `index` has to be in the range `[1, length(array)]` (responsibility of the consumer)
2. If consumer calls the method adhering to (1), then after the method call the following holds:
   `arr[index] < arr[index+1] < ... < arr[length(array)]` (responsibility of the supplier)

Pre- and post-conditions

```
% Sorts the array in ascending order starting
% from index
%
% PRECOND: 0 < index <= length(array)
% POSTCOND: arr(index) < ...
% ... < arr(length(array))
function arr = sortArrayFromIndex(array, index)
```

- Responsibilities of the consumer are method *pre-conditions* ("Requires")
- Responsibilities of the supplier are method *post-conditions* ("Ensures")
- (PRECOND, METHOD) $\Rightarrow$ POSTCOND

Violating pre-conditions

- As a supplier, if the pre-condition is violated, you are not responsible for what happens
- In practice you should crash the program execution, as the mistake is in the logic

```
function array = sortFromIndex(array, index)
  assert(index > 0 && index <= length(array));
  ... % do the actual sorting
end
```

- In R: `stopifnot`

When to use pre- and post-conditions

- If you cannot handle a possible parameter value, you should declare the accepted range as a pre-conditions
- Post-conditions are often stated in a more informal manner in the method documentations
- Document post-conditions when doing more complex programs, and when you have problems finding bugs

Example: isBurned function from the current exercise

Create a function that checks whether sets of coordinates have been burned by any of the current fires.

- Depending on how the forest fires are stored in your application, this function should at least take as input:
  1. A matrix with all sets of coordinates to be checked.
  2. The matrix with information on all current fires, including the coordinates of their centers.
- The output of this function should be a vector, signalling for each set of coordinates whether this point has been burned (TRUE) or not (FALSE).

(PRECOND, METHOD) $\Rightarrow$ POSTCOND

How do we know that METHOD ever terminates execution?
How do we know that METHOD does what it's supposed to?