*Question 2*

The biggest disadvantage of this implementation is that it will be almost impossible to implement the checks that are needed to be done in Runtime, since the call for invariant() is injected during the class load. So any JRE states are not available since the code isn't executed yet. We can't check who called the method if it hasn't been called yet!

Moreover, the checking of the conditions for a valid call to *invariant* must now be checked inside *invariant* itself, after it was already invoked. This means that invocation will always occur, no matter if the conditions are correct. Furthermore, the responsibility for the checking of these conditions is now upon the *invariant* method and not upon the framework, which is not the initial purpose of the *invariant* method.

It would be possible to simply inject the entire check() bytecode into the end of each method, but this practically saves nothing (besides several register assignments and “goto” instructions). It would also be extremely inconvenient for maintenance and future modifications, and the injection will have to take into consideration existing labels in the checkable method bytecode (e.g. local variable names or method calls collisions).