Warning List.
Departamento de Computação – UFSCar.
José de Oliveira Guimarães.

This text presents a list of warnings that a compiler of an object-oriented language could issue. The list is general and almost every item applies to almost every OOL.

1. Private instance variable declared but never used. Of course, this deserves a warning only if the instance variable is private. And in Simples all IV are private. Note that by "used" we mean used where an expression is expected. It does not count if the variable is used in the left-hand side of an assignment or as a parameter to `read`.

2. Private instance variable used but never instantiated. By "the variable is used" we mean it is used where an expression is expected. Notice a variable may be instantiated in an assignment or in a `read` statement (in Simples).

3. Private instance variable used in only one method. It should be replaced by a local variable.

4. Instance variable instantiated in more than one method. A good practice is to instantiate an IV in only one method. However, in some situations it is inevitable to assign values to an IV in several methods.

5. Private method is never called.

6. Method does not use any instance variable and it does not call other class methods using self. In Simples, this mean the keyword "`self`" should appear at least once in each method.

7. Method is declared with a return type but it does not return a value.

8. Local variable or parameter declared but never used.

9. Variable is instantiated once with a literal value. This warning should be issued if a variable `x` is instantiated with, say, `0` in an assignment and this is the only place in which `x` receives a value. Then `x` could be replaced by a constant.

10. Instantiation of a parameter. A parameter should never receive a value in an instantiation. This is bad practice. Instantiation includes assignments and call to `read`.

11. Class is never used.

12. Unnecessary test to `true` or `false`. Tests like "`x == true`" are redundant. This test can be replaced by "`x`". A test `found == false` can be replaced by `not found`, which is clearer.

13. Method is too big. This warning should apply to methods that have more than, say, N statements. N could be used defined.

14. Class is too big. The compiler should issue this warning if the class defines more than M methods

or K instance variables. M and K should be used defined. Of course, inherited methods and instance variables should not be considered.

15. To a variable is assigned a value but it is never used. This occurs if the variable is never used after the instantiation or if it receives another value without being used between the instantiations.

16. Literal number used outside `const` declaration. Only `0` and `1` should appear in the program outside a const declaration. Since Simples does not have constants, this rule does not apply to Simples compilers.

17. Possible division by zero and division by zero. There are two flavors:

```
b = b/0;
a = 0;
c = search(x);
b = b/a;
```

The compiler should issue two warnings "Division by zero" and "Possible division by zero" when compiling the code above.

18. Method defined in all subclasses but not in the superclass. If this occurs, the method should at least be defined in the superclass as abstract.

19. Instance variable defined in all subclasses but not in the superclass. It should be defined in the superclass.