Prof. Dr. A. Poetzsch-Heffter
Dipl.-Inform. Kathrin Geilmann

University of Kaiserslautern
Department of Computer Science
Software Technology Group

Advanced Aspects of Object-Oriented Programming (SS 2010)

Practice Sheet 2 (Hints and Comments)

Exercise 1 Happy Birthday

a) The problem with the given implementation is, that the current age of a `AgePerson` is used in the method `hashCode`. That changes the hash of the instance stored in the `HashMap` of the `AgeManager`.

The documentation of the interface `Map` states:

"Note: great care must be exercised if mutable objects are used as map keys. The behavior of a map is not specified if the value of an object is changed in a manner that affects `equals` comparisons while the object is a key in the map. A special case of this prohibition is that it is not permissible for a map to contain itself as a key. While it is permissible for a map to contain itself as a value, extreme caution is advised: the `equals` and `hashCode` methods are no longer well defined on a such a map.

b) To get rid of these problems, don't use mutable keys in maps. In general it is a good idea not to use mutable data to calculate hashes, instead base the hash calculation for a class on its immutable attributes.

c) In the method `AgeManager.add` a anonymous inner class is created. The parameter `birthday` is used in its initializer block. The Java-Specification states in that case, that this parameter has to be final. (see JLS 8.1.3). This is because the instance of the inner class must maintain a separate copy of the variable, as it may out-live the function, in which the instance is created.

Exercise 2 Source Compatibility of Java-Packages

Package `util` The modification breaks source compatibility, because if someone implemented the `List` of the unmodified package, and tries to compile his source with the modified version, the compiler will complain about the not implemented method `m`.

Package `p` This modification is safe. `C` was declared final, so there cannot be any subclasses of `C` and `m` was declared `protected`, therefor the only code that can call `m` is inside `p`, the change of the parameter type is safe because it only affects the package `p` but nothing outside it. The changes from final to non-final and protected to public are safe as well, as it just widens the accessibility, which is no problem as there have not been any subclasses that may clash with the changed declarations.

For more information you may look at `http://softech.informatik.uni-kl.de/Homepage/SourceCompatibility`.