3. If we had to implement differently the Stack class without changing all the methods, we would have used generic types instead of hard-defined types (i.e. int). It would avoid a lot of changes, and the object would be more flexible in the end.

The only difference would be in the types and not in the global architecture.

4. We wouldn’t need to rewrite all the tests. The only one using a strongly degined type (head(), push() or some constructors) would be rewritten.

5.

9. We won’t need to rewrite all the tests. The only tests that are needed are for dequeue(), enqueue(), head(), back() (in order to check the algorithm for these 4 methods), and the constructors : Queue(), Queue(int), Queue(int, int[]).

Some methods are exactly the same as the ones in the Stack class but used in the Queue context, calling Stack objects : therefore, the tests that imply Stacks are useless.

10.

11. We have reused code from the Stack class to develop the Queue class, and from the Stack\_tests class to develop the Queue\_tests class.

Another solution would have been to make the Queue class extend from Stack: this way, the code wouldn’t need to be reused, the only thing that we could refactor would be the 2nd array inside the table to manage the input and the output. The new functions would have to be coded (head, back) and the old functions to be refactored (push and pop become Enqueue and Dequeue).