ERRATA 514: Section 14.3 mailbox type mismatch

14.3.5 get()

Change:

If ~~there isa~~ the type ~~mismatch between~~ of the *message* variable is not equivalent to the type of ~~and~~ the message in the mailbox, a runtime error is generated.

14.3.6 try_get()

Change:

The try_get() method tries to retrieve one message from the mailbox. If the mailbox is empty, then the method returns 0. If ~~there isa~~ the type ~~mismatch between~~ of the *message* variable is not equivalent to the type of ~~and~~ the message in the mailbox, the method returns –1. If a message is available and the message type ~~matches~~ is equivalent to the type of the *message* variable, the message is retrieved and the method returns 1.

14.3.6 peek()

Change:

If ~~there isa~~ the type ~~mismatch between~~ of the *message* variable is not equivalent to the type of ~~and~~ the message in the mailbox, a runtime error is generated.

14.3.7 try_peek()

Change:

If ~~there isa~~ the type ~~mismatch between~~ of the *message* variable is not equivalent to the type of ~~and~~ the message in the mailbox, the method returns –1. If a message is available and its type is equivalent to the type of the *message* ~~type matches~~ variable, the message is copied and the method returns 1.

14.4 Parameterized mailboxes

Change:

This is a very powerful mechanism that, unfortunately, can also result in run-time errors due to type mismatches (types not equivalent) between a message and the type of the variable used to retrieve the message. Frequently, a mailbox is used to transfer a particular message type, and, in that case, it is useful to detect type mismatches at compile time.

The only difference between a generic (dynamic) mailbox and a parameterized mailbox is that for a parameterized mailbox, the compiler ensures that the calls to put, try_put, peek, try_peek, get and try_get methods ~~are~~use argument types equivalent to~~compatible with~~ the mailbox message type, so that all type mismatches are caught by the compiler and not at run-time.