**PODCAST: AFRICAN DEVELOPER PODCAST**

**EPISODE 3: HANDLING THE IMPOSSIBLE**

Have you noticed that sometimes other people can detect that thing aren’t well with you before you’re aware of the problem yourself? This can happen in our personal lives as well as our careers. Such a revelations can be surprising and we might even dismiss it.

Ok! Back to software development! If something is starting to go bad with one of our programs, we are usually not the first person to notice it. When that bug report is made it can catch us by surprise and we can even swear that it “impossible”

It’s easy to fall into the “it can’t happen” mentality. Most of us have written code that doesn’t check certain conditions or we can make assumptions that the programme will mostly be in a predictable state. With quality in mind, we can have a bunch of tests in place and at deployment we can be confident the code we have deployed wouldn’t fail under any normal conditions. But the “impossible” happens a weird error happens and your left thinking “this is impossible”.

When errors happen something beautiful happens! I’m not saying errors are beautiful, but as a software developer all your career you will be dealing with errors one way or another. When errors happen clues are usually are left behind inform of a stack trace, a log entry or a notification about what has really happened. You could convince yourself that the error can’t happen, and choose to ignore it. But ignoring errors no matter how small can be very dangerous since some small errors can manifest and if their root cause is investigated can lead to finding really nasty bugs.

**Crash Early**

One of the benefits of detecting problems as soon as you can is that you can crash earlier. And many times, crashing your program may sound really bad but sometimes it might be the best thing you can do. The alternative may be to continue with execution. But you might end up writing corrupted data to some vital database.

The Java language and libraries have embraced this philosophy. When something unexpected happens within the runtime system, it throws a RuntimeException. If not caught, this will percolate up to the top level of the program and cause it to halt, displaying a stack trace.

You can do the same in other languages. If you don’t have an exception mechanism, or if your libraries don’t throw exceptions, then make sure you handle the errors yourself.

Clearly it is sometimes inappropriate simply to exit a running program. You may have claimed resources that might not get released, or you may need to write log messages, tidy up open transactions, or interact with other processes.

However, the basic principle stays the same—when your code discovers that something that was supposed to be “impossible” just happened, your program is no longer viable. Anything it does from this point forward becomes suspect, so terminate it as soon as possible. A dead program normally does a lot less damage than a crippled one.