Jack O'Neill

Mobile App Development I

Dr. Haim Levkowitz

11/29/2021

Inception – My First Experience with Mobile Development in Kotlin

Programming at the behest of another typically involves the same routine. The commissioner will outline their expectations for the program and what specifications it must meet. Given these instructions, the programmer's job is to produce a program that is tailored to meet the needs and expectations of the commissioner. However, what happens when the project you have been asked to produce is open ended? This paper will explore the natural outcome of being given the freedom to work on whatever one may like, for better or for worse. It will also talk in detail about my perspective on programming in both Java and Kotlin, what that taught me about programming, and what that taught me about myself.

My initial concept for Inception was to make an application that I myself would use in my daily life. I wanted to develop a task and idea organizer for creatives, as most calendar apps and similar programs are business oriented. Peradventure I finish this course, I thought, I should have something that is useful to me as a result of my hard work. However, my ambition outpaced my abilities. While the final iteration of the program is functional, it has not achieved all of the admittedly lofty goals I had originally set for it. Exacerbated by my work on this project being solo alongside the fact that this is my first time developing in Kotlin, most of my time was spent debugging and revising the program to become something more sensible yet still functional.

All of this agonizing and tribulating over the minutia that could have been implemented or features left untouched initially left me with one simple burning question; why not avoid the problem of failing to fully realize such a large-scale product by making something much simpler? For my case, I attribute my reluctance to have taken that path to two main factors.

First among the factors that I attribute to the inflation of my project's scale is the scalar issue itself, that is, one of quantity. During the inception of Inception, my plans were not initially so lofty as they appear in the project proposal. However, as the time to submit the proposal approached, I had doubts that perhaps my program might be lacking. Sure, this was something I was making for myself to solve problems that I and people like me encounter on a daily basis, but what of the outsider's perspective? Will the offerings of the application seem paltry in comparison to competitors in the same field? It was by that metric that I gauged the scale of my program, but it was also a metric by which I overinflated its scale.

The second factor that contributed to the bloated objectives of my project proposal was the factor of relativity. It is fine and well that my program operates as expected, but what is there to be said about its capabilities relative to those of my colleagues or direct competitors? The main incident I can draw upon that encapsulates this factor was the day of in-class pitches. Pitching in front of the class was a great way to get constructive feedback and criticism for my project. Listening to the proposals of other students, however, set my mind in motion, not in a

direction that would refine what I already had but one that would tack on features or concepts I thought might be helpful.

Unfortunately, the time for all things must come to pass, and along that same vein, the time allotted to properly execute my lofty goals was finite. In the end, I did not implement every feature that I expected to load my program with; in fact, the final program was shaved down quite a bit from its original implementation. The program in its current form is functional and accomplishes the tasks that it has currently implemented correctly. What exists now is not a complete vision of the original proposal, but I would call it a complete program in the sense that if this were provided to a user, they would be none the wiser regarding its initial lofty goals.

The current implementation of Inception has been boiled down into two components: a task tracker and a hyperlink repository.

The task tracker allows users to track their current tasks, their descriptions, when they are due to be completed, their level of difficulty and their completion status. Adding new tasks is done through a separate activity that offers three difficulties, a description field, and a calendar whereby the user may select a date for the task's completion.

The hyperlink repository is one of my more interesting ideas. The thought process goes something like this; "if the user is capable of opening the link in a web browser, then they are capable of storing it in the repository." This could by just about anything that might come to the imagination of the user. One could, more obviously, store a link to a YouTube video or something of the same variety. Less obviously, however, the user could easily open a file in local storage from the link and play it in the web browser, so long as the file type was compliant. For example, I know from experience I can play a video file from my phone on a web browser, and it works just fine. In the link field of the browser, I can find displayed the file path to the video. This should be able to be utilized as well.

With this all stated, it should be clear why I am labeling my project as a "complete program." Although this program is not exact to its original project proposal, what features it does include are themselves complete and functional to the point where, as previously stated, the uninformed end user would not know any better.

On the subject of features failing to make it into the program because of time constraints, I would consider it prudent to mention a factor that weighed heavily on the development speed of this project. There is one standout feature of mobile development that is nearly inescapable for android developers, that being the Java family of languages.

Excluding something like Scratch, because I'm not going to compare block coding to typed coding, Java was my first true experience with programming. That being said, I personally believe that Java is one of the primary reasons that I was slow to latch on to programming. As a newcomer, everything that Java expected of me, not as an end user but as a programmer, was honestly daunting. Want to make a data structure? How about you make an entirely separate class file for it? Better yet, want to use a function from an imported library? Maybe Java will make you qualify it with a try/catch statement; this is not optional, by the way. The program will not compile unless you follow the safety rules. Peradventure that, God willing, I managed follow

all the rules to a tee, getting Eclipse to work properly was often a dice roll for me in those older days.

The Java way of doing things, in my perspective, slows things down. Getting so caught up in putting everything where it belongs, festering over whether your class inheritance hierarchy is correct, and lining your code in enough bubble wrap to line a small shipping container clogs the programmer's progress when the same functionality could have been executed in fewer lines of code. Sure, maybe your program *looks* neat under the hood, and perhaps it is safer to have mandatory try/catch statements, but shouldn't I, as the programmer, be able to choose whether or not I want to tap into those benefits? What if I just wanted to prototype something? What might take me 5 minutes in Python would likely take 10 minutes in Java due to the extra formalities that Java expects of programmers.

When I first arrived at UMass Lowell, I started learning in C. It was initially difficult to understand, but not as difficult to execute. In fact, a strong basis in C has informed my understanding of most other programming languages. The best part about C is that it doesn't babysit you. My father once told me, in regard to the C language, a quote from one of his old professors. It went something along the lines of, "C gives you just enough rope to play hangman with." I consider it a gross coincidence that the Computing 2 Lab final project is to program a hangman game in C, although I digress.

Ultimately, what I'm trying to say boils down to a very simple idea, the idea being that the best programming language for me is one that gives me the freedom to do something that is informal, hacky, or even poorly formulated. If a language does not even grant me permission to compile my program, not for the reason that it is uncompilable but that it is "not guarded enough to compile," I can't help but feeling trapped. I would gladly take the freedom to break my program over safety any day.

So, where am I ultimately going with this rant about the Java family? As stated previously, and as you might be able to guess, this class constituted my reintroduction to an old frenemy of mine, Java. For the first programming assignments of the class, I worked entirely in Java since it was the language with which I was most familiar. However, all of my initial gripes with Java slowly crept back to the surface as I delved deeper into the projects.

One tribulation in particular sticks out clearly in my memory. For the weather app, we were required to tap an API. All of my code, at least in any other language, was syntactically and programmatically correct. However, Java told me there was not enough bubble wrap on my code. First, the familiar try/catch requirements; if I don't qualify my code within that context, it refuses to compile for what I can assume are purely political reasons. Second was something new to me in all of my time programming; making the API call was rejected by Java in the form I put it, not because it wouldn't theoretically function, but because it would run on the same thread as the same program. "If your internet dips," the error log remarked, "your program may stall. Move the code to a new thread or you cannot compile this." In the end, I had to arbitrarily wrap the code in a function to run it in a separate thread. I'm not saying that the advice was bad advice, on the contrary, that was great advice and something I will keep in mind throughout my future programming endeavors. That being said, a suggestion like that should only be a compiler warning, not a compiler error.

Since I went on for all of those paragraphs about Java, you may be surprised to hear that I didn't even code my final project in Java, at least not technically. When I went to create the repository for my final project, android studio gave me two options for developing programs, those being Java and Kotlin. In an effort to dodge Java, I decided to bite the bullet and learn a new language, Kotlin

What I didn't realize is that what I had made was a false choice. Kotlin as far as I understand it appears to be one giant compatibility layer on top of Java to make it simpler and less painful for mobile development. This was great in the sense that the language I was working in was now decreased in complexity, but not so great in the sense that it was all still Java under the hood. If you write Java code into a Kotlin project, Android Studio responds with the suggestion, "This appears to be Java code. Would you like to use the conversion script to change it to Kotlin?" That was the moment that pierced the veil for me and brought me face to face once again with Java.

Ultimately what I was left with was working with a slightly simpler Java language, masked by a new syntax. That meant more time learning and troubleshooting, and likewise, less time actually implementing the features I wanted to. Even if the program was in Java from the start, I set my goals too far off, and declared as a 1-semester project what was very likely a 2-semester project.

Despite the pessimism baked into my description of the development workflow, I am ultimately happy with what I have accomplished against some pretty stacked odds. In the face of a time crunch against an out of reach goal set to a foreign language with more so alien conventions and quirks, what I have managed to produce is impressive to me. I ask not for your compliant agreeance with my own sentiments, only your understanding of them, and hope that, in some capacity, you can appreciate what I have accomplished for what it is while simultaneously learning from my mistakes.