experience of Kotlin

I had a positive experience overall and would recommend Kotlin to android developers. Kotlin has an active (but smaller than Java) developer community but generally it was still easy to find up-to-date examples and documentation. It has many similarities with Java so it wasn't hard to learn and many of the IDEs that support Java (e.g eclipse/android-studio) also support Kotlin. According to Jetbrains, companies that are using Kotlin include Pinterest, AWS, and Netflix, and Google provided official support for Kotlin in may 2017 (according to indeed.com, this significantly increased the number of job listings that mention Kotlin) so it seems to be increasing in popularity. Android Studio can convert parts of Java code to Kotlin so it's easy to migrate projects written in Java and it's possible to use existing java frameworks and libraries which makes migrating to kotlin easier. The Gradle build system, and IDE features like auto-imports (Alt-Enter) and tab-completion all work with Kotlin. Kotlin has many additional features that java doesn't (e.g lambdas, implicit casting and type inference) which made it easier to write shorter code. This can also make the code harder to understand, (e.g . the safe call operator ? used with let and run can be used to avoid null checks but if you need to check multiple variables and chain them, then it could be confusing, and it would be clearer to do the null checks with booleans inside an if statement). In Java you always need to declare the return type of a method but in Kotlin you don't need to when you use expression syntax and you can leave out the return type for functions that do not return anything (which need to be declared void in Java) because unit is the default return type. Kotlin has a nothing type which makes it easier to find bugs and it is used for unreachable code warnings in Android Studio which java doesn't have. Kotlin has type safety with nullable types and this helps prevent null pointer exception errors. I prefer Kotlin's for loop syntax over Java's because it's more concise and flexible - any progression or range can be inside the (), for example, I've used for(i in 0 until num) which is the same as for(i=1,i¡num,i++) in Java. the 'until' keyword makes the range exclude the end (num) and this is the same as range(0,n) in python which excludes n by default, this makes the for loop in Kotlin more idiomatic and readable. ranges that exclude the end are more common because of 0 based indexing and terminating early is less serious than a buffer overflow. Kotlin is still newer and less commonly used than Java and some applications still do not support Kotlin, e.g the listings package for latex code formatting does not support Kotlin but supports Java.

bonus features

The bonus features I've implemented are: 1. five difficulty levels 2. a scoring system 3. displaying the time it took to guess the song

features in the design document not implemented: 1. I said there would be a 'words transfer' feature in the design document but I didn't do this New features: 1. I said the song would be randomly picked but I've added a start acreen listing the songs (guessed and unknown) so that the player can pick a song and see the songs they've guessed correctly each time on the start screen 2. a list of 'guessed' songs with the date guessed

acknowledgements

I mainly looked at blog posts (e.g medium /jetbrains), the lecture slides, documentation from developer.android.com, kotlinlang, and stackover-flow here are some specific links: https://developer.android.com/reference/android/location/Lo
https://developer.android.com/studio/write/layout-editor.html https://www.programiz.com/
programming