# Written Exercise – 5 minutes

1. Go through the following list of example concepts. (They are quotes from a study about how developers solve a bug.)
2. In your own notebook, for each numbered concept, write down:
   a) Whether the concept is Surface or Depth
   b) The type of the concept.

Surface concept types:
- explanation
- preference
- opinion
- scene-setting
- statement-of-fact
- generalization

Depth concept types:
- inner thinking
- emotional reaction
- guiding principle

1. Because a lot of times, like when you design UI, user interfaces, sometimes things can be too simple. And you don't get the end result that you're looking for. Sometimes things are too simple.
2. I will not do it that way because I have this body of code. I will do it only the way that I think it should be done for this particular domain and this particular problem. I'll pull it out of the toolbox when I have the same thing to solve, but I'll build something new and put it in the toolbox if it's different.
3. Because I don't want to reinvent the wheel if the wheel's already coming … I'm not going to waste hours and hours of development effort to try to crank out on-screen keyboard for Android if I already see it's on the roadmap.
4. What we do is we get the tasks – we kind of get the high level tasks of where we want to go, like a end result. Then we kind of set milestones in between. Like by this date, we should have at least this much done. By this date, we should have even more done, even more so, until we get to completion.
5. When we installed on Linux there were issues related to environment readiness, and the error messages were not really proper. So it was quite difficult to debug … Error message would not really inform you; it was not really helping me to figure out what went wrong.
6. Architecture just kind of – it's really based on – one, it's like language, which computer language we're going to be using. But when you're dealing with like – with Apple, of course, you obviously have to use Objective C.
7. What I've done on this project, is like take a problem that you're not sure how to solve in this particular language, and solve it in a language that you know how to solve it in. And then just kind of, you know, write a smaller program that does it in the language that you know how to do it in. Then on the side of that, try to write it in Java.

the answers are on the next slides

1. Surface – opinion: Because a lot of times, like when you design UI, user interfaces, sometimes things can be too simple.  And you don't get the end result that you're looking for.  Sometimes things are too simple. (IY: It's a generalization, not tied to a point in time. I tried to see if there was implied emotional frustration, but it wasn't clear enough. That last sentence is an opinion, so that's what I decided this was.)

2. Depth – guiding principle: I will not do it that way because I have this body of code.  I will do it only the way that I think it should be done for this particular domain and this particular problem.  I'll pull it out of the toolbox when I have the same thing to solve, but I'll build something new and put it in the toolbox if it's different. (IY: It is a description about how they make decisions given certain contexts, which is the very definition of a guiding principle.)

3. Depth – guiding principle: Because I don't want to reinvent the wheel if the wheel's already coming … I'm not going to waste hours and hours of development effort to try to crank out on-screen keyboard for Android if I already see it's on the roadmap. (IY: Again, this is a rule they follow given certain circumstances.)

4. Surface – explanation: What we do is we get the tasks – we kind of get the high level tasks of where we want to go, like a end result.  Then we kind of set milestones in between.  Like by this date, we should have at least this much done.  By this date, we should have even more done, even more so, until we get to completion. (IY: Speaker explaining to the Listener how they plan a project. It would be nice to hear inner thinking, or history that got them to this approach.)

5. Depth – (implied) emotional reaction: When we installed on Linux there were issues related to environment readiness, and the error messages were not really proper. So it was quite difficult to debug … Error message would not really inform you; it was not really helping me to figure out what went wrong. (IY: it reads like an explanation of what happened, but there is a layer of emotion here. The Speaker may not admit to the emotion, but they sound a bit frustrated. With implied emotions, you can use the rest of the transcript to gauge what the person is like, emotionally, for contexts like these.)

6. Surface – explanation: Architecture just kind of – it's really based on – one, it's like language, which computer language we're going to be using. But when you're dealing with like – with Apple, of course, you obviously have to use Objective C. (IY: The first part is explanation, or statement of fact, either one. The Speaker is explaining to the Listener, about how architecture is linked to language. Getting their inner thinking would help here.)

7. Depth – inner thinking: What I've done on this project, is like take a problem that you're not sure how to solve in this particular language, and solve it in a language that you know how to solve it in. And then just kind of, you know, write a smaller program that does it in the language that you know how to do it in. Then on the side of that, try to write it in Java. (IY: They use the phrase "you know" in two different ways, so it helped me to recite it aloud. This is at a time and place, so not a generalization. It's their thinking as they began this project, and the way they say it could be extended to be a guiding principle. So, either.)

surface vs depth is the important part

which type of surface or depth is less important

if it's depth, and you label it as inner thinking but it's really a guiding principle, it doesn't matter … the concept will get put into a pattern with similar concepts no matter what its type