Okay, hey, this is Taylor. We're whispering and talking about the Space Time Java application with Dr. Jonathan Bennett. It's Friday, July 26th about 1 30 PM. Perfect thing to be doing together on a Friday afternoon in the summertime. So we're looking at here the class structure and we've identified the first class here, drawing panel highway and how it connects to what we saw when we run the application in the screenshot. And we've been looking through some of the source code, getting familiar with IntelliJ, looking at the mini map, the scroll bar, the warning symbols, the code recommendations for code styling, and I think our next dive is going to be into the Java provided classes. I think I just wanted to cover that for a few minutes so that you begin to understand the class hierarchy and that'll help identify how the different application classes render into the user interface. So let's go up to the top of the file and look for the the class definition line. You can see in IntelliJ, there it is line 26. So public class drawing panel highway. That's its name and then it has a keyword extends and a keyword implements. And let's hover over. I say right click on the word drawing panel. So it extends drawing panel and right click and let's choose go to. And we'll do. Implementations the third option go to implementations. Alright, and let's just choose the first one. Cool. Oh, okay. So with that go to you can use it to navigate around to other references and usages in the code. You don't always have to search on the left to to find the source file. You can follow references automatically from the lines of code by right clicking. Okay. So it looks like there's a sister here. Drawing panel highway is like drawing panel diagram in that they both extend a drawing panel. Oh, panel is its own class exactly. So let's open the drawing panel class. So you can see here on line nineteen, it's declared abstract public abstract and that means that it can't actually exist. But everything in here is available to both the drawing panel diagram and available to drawing panel highway. It's like shared code. So this code. So. But in tandem with the the other code that that kind of builds on it, it can draw stuff. Mm hmm. Yeah. So the word abstract. Take that. It's the advantage of having a type of class that's abstract. It. Let's see here. That's a fantastic question. It means that it can never be directly instantiated. So it it marks it as this is reusable code. This is only code that's designed to be reused in other places. So if you in the in your windows task bar, um I see the intelligence active and then there's a zoom icon and then there's a little guy with the red nose. His name is Duke. So let's click Duke and it'll bring the app back up. Little guy with the red nose. Are you down here? Yeah. Yeah, let's let's pop that application back up. This? Yes, please. Didn't even recognize it as a guy with the red nose. Okay. So, I see on the left hand, we have the highway and the diagram and they to me have a lot of the same visual attributes. They have a Y axis and an X axis that's labeled. Right. And so that I imagine is what would be common between them that's gonna be declared in the drawing panel abstract. And maybe they're vertically aligned that they have the same width to their extent and so maybe by. The dimensions are defined in the in the abstract part of the code so that that pattern gets shared right and always keeps them vertically aligned. Yes. Yeah, there's a there's a key principle in modern software design. It's abbreviation is DRY. You gotta stay dry which means don't repeat yourself. Don't ever copy paste code in multiple locations cuz if if it has a bug, you gotta fix that bug in all those locations. Instead, put it in a common single implementation. Stay dry. Cool, right? So, if I look at this drawing panel here, hopefully, this is not a 550 line. This is a little bit more manageable. 150 lines. Yeah. And so. Yeah, it looks like code for dragging and scaling and the mouse interactions. Okay. Okay. So, there's no physics here. This is this is graphics and yeah. Making things pop up. Yep. At the place on the screen where you want them to pop up. Exactly. So, let's talk about, I see functions here from line ninety-three to 105. This is a super common thing that drawing systems have is you've gotta translate your coordinate system from the physics coordinates into the on-screen graphical coordinates. Yeah. So, let's say for instance, you resize to the window. Made it full screen or made it half screen. This is the code that's gonna know how to do that translation no matter what size your graphical window is. So, So, this is not the code that actually does the drawing though I would think. This is the code that tells some other code. This code is gonna be used by the drawing code. So, for instance, line ninety-three, right click the function name X to Pics and we'll use that go to or yeah, I'm sorry. Let's use find usages above go to. Click that. And then on the in that bottom window on the left method and that has the usages or usages of that's used fifty-three times. That's super dry. Is that that function is very popular and reused in a lot of different places in the code base. And so, if you expand that tree, that usages, it'll give you a listing of all the different locations who calls it. Of course, the diagram, the highway, change events, clocks, and events. Okay. So, drawing pin effects. So, this implements a mouse listener, mouse motion listener, and it extends. This is some built-in Java class, I guess. Draw stuff. You've got it exactly right. And that's not something that we're gonna mess with it, right? I mean, that's just and that's not something that this programmer had to write. He just, that's a built-in feature of Java that anybody who uses Java just calls that. Exactly right. And so that Java X swing all of the different components and the buttons and the window frames and the text boxes, all of the essential components of how you build a user interface, that is what's provided by Swing. And so, when you're writing an application, you just reuse all that stuff to build the user interface. You don't need to program all the low level. You don't have to tell it how to make a button. Don't tell it how to make a button. You just use J button. You don't have to tell it how to make a window frame. You just use J frame. So, somewhere down here, I don't think he's got buttons on his application but he's got window frames. So, there's probably a J frame. Yup. I think I think I saw it in the app itself in Space Time app. It's a it's a pretty common thing to do for your main class, your your application class to extend J frame. Cuz that just always makes it pop up in a window which is how most 99% of applications work. They just pop up in a window. Okay, let me go back. I'm gonna go back over there. Okay, extends J frame. Yup. But there still must be a a manned line in the script that tells it pop up. Put a window and put it at this place on the screen. Yeah, line 1036. Set visible true. Okay. Cool. So, there's another aspect of this that I wanted to talk about while we're down in that drawing panel abstract. And that's it's a similar idea to abstract classes and they're called interfaces. So, up at the top, IntelliJ has highlighted that for you but you can go back up to line 19. And after the extends, it also has implements. So, this this one implements mouse listener. So, that's a pretty good tell to me that it will be able to respond to when the user moves the mouse around and clicks and drags and everything. And so, somewhere down, I mean, I know, continually checking what's the position of the mouse, what's the position of the mouse, and is you're clicking at that position. I can move the mouse but not click on it. So, And so that that mouse listener is what connects the computer and the user who's actually moving the mouse around the screen. It's the connection to send those that activity into the software application. So, Java has a area that's essentially the position of the XY coordinates and pixels of the mouse and it'll it knows where the mouse is. So, if I'm dragging something. Yep. And it it reports those variables every single time it even moves a pixel, it's this high-speed reporting interface where it's constantly sending that information to the application. That's what mouse listener does. Exactly. As well as detecting clicks on that little. Yep. And so, we were talking earlier about the how Java is write once, run anywhere. It works on Windows. It works on Mac. It works on Linux. What mouse listener does is it generalizes everything so you don't have to worry about all the operating system specifics. Whether you're using a track pad or a track ball or an actual real mouse. All that's abstracted away no matter what type of hardware you're on. It's just a mouse listener. Hardware. There's some other software in my computer that makes the mouse move around but that's that's independent of the Java. Java just doesn't. There is the mouse. Yep. That's what drivers are for, right? I have some driver for. Yup. Tells my computer what to do when I move my. That's right. And so, it would be so hard if all, if every single application developer had to worry about those drivers and that's how it used to be in the IBM personal computer days. The 286 to 386. Every single app that shipped had to also ship with like sound drivers and graphics drivers. And Java really was invented to fix that. Make things more scalable. Make applications faster to write. A little bit of history. Appreciate the lesson. So, before we leave off these the mouse listener, what I wanted to do was highlight the language syntax in Java of how it does it. And it does it through functions. So, if you scroll to the bottom of this source code, I think I saw the functions. The drawing panel? Mm hmm. Yep, the bottom of drawing panel. I believe has the mouse function. There they are. So, mouse entered, mouse exited, mouse pressed, mouse released, dragged, and moved. And those functions pass in a parameter called mouse event. So, I'm curious, let's go look at what mouse event has available in it. So, if you right click, go to type declaration. Maybe go to declaration or usages. Alright. So, you've now descended into the Java code. This is another thing that's really cool about open source and about IDEs is you can actually go read their code. And in here, let's look at some of the variables that it provides. So, I would scroll down a little bit. There's some integers. So, I'm trying to figure out how would I know like what what position what position is the mouse in? So, I'm assuming these integers are the number of a pixel. These integers are These are a shortcut to make to make it easier to understand the code. So, so I wouldn't worry too much about these integers. All they're really making is like line 240. It lets you use the string mouse underscore exited in your code instead of needing to remember like the number 567. Wow. It's just making it more syntactically accessible to you as an application developer. So, let's keep scrolling down. Look for some functions. Button one, button two, button three. I don't know that my track pad has a button three. Cool. There you go. So, whenever that function is called, that mouse clicked, it's going to pass in one of these mouse events and in the mouse event, you can just call event dot x and you know then the x position of where the click was. Okay. And so, say dot x. So, that's so x is the name of a variable in the class called mouse event. So, I can if I wanted to refer to that variable that I understand you quickly, I would say mouse event dot x. Yes. That's the name of a variable. Yep. That's provided by the mouse event class. So, with that understanding, I think we're ready now to go back up to the the drawing panel highway which is four source files to the left and let's look for those mouse event functions here. So, I've looked at the panel highway. So, that's giving me the name of this. Mouse wheel listening. Okay. Now, it's already, what already knows that it's extending a drawing panel. And that it's implementing the mouse wheel listener. Yep. As well as. Yes. And it's it's also implementing the mouse event listener because it's parent class drawing panel implements that mouse event listener. So, this guy can handle clicks as well as wheels rolling the mouse wheel up and down. So, by using inherit class inheritance with extends and interfaces with implements, you can blend together this functionality in lots of different creative ways. So, let's look for. Okay. Sorry. Quick question. We already said up here that we are implementing mouse wheel listening. Why do I need to have a line that says to add the mouse wheel listener? Yes, that's what we're about to get into. So, let's right click this guy and we'll go, we'll go to definition. Say declaration or usages. Or. Back pad is not tracking. So, let me try this again. Okay. Yep. So, this is in the swing library and this is just something that you've gotta do to basically connect the dots between your class and the swing library. In Java, they it's just the way they designed it. They they wanted you to be explicit about making those connections, not try to have it be too automatic for you. So, it leaves the burden on the programmer to actually do the final hook up which is what that add mouse listener does. So, it's just sort of the way it's the way it's done that address listener has to be called or else it won't receive the events. Okay. Great. Well, that's just a built-in Java feature. Exactly. Very nice beta values. That's that is some function or subroutine that's defined those. Mm hmm. Yeah. So, I could navigate and look at it if I need to. Okay. Alright. So, let's look. Let's let's search through here for the the mouse clicked function. So, you could probably do control F and search for mouse click. There it is. Line 234. And so, your question was How do I know If I call mouse event.x, that's gonna know the x pixel of where the mouse was clicked. So, I think we should read this code and see if in somewhere the author calls event.x. Do you wanna if I take a real quick call? I'll be right back. Yeah, no problem. Good afternoon, Taylor Brockman. Hey, Becky. Good. How are you? Um I am typically on a I'm typically on a customer call Monday. Um I could leave II could leave a lockbox key for him. I will set it to one two seven nine. Thank you, my friend. Have a good night. Bye. Okay, I'm back. Thank you. See that it's it's it looks like this little part of the script is simply checking if if a mouse was clicked and if it is, it's it redraws everything. Mm hmm. Turns off whatever was already drawn and draw something new. Yep, I think it's that that's it. I agree. So, repaint some other code that actually does the drawable. Mm hmm. Yeah, so I would search the source code for paint. And I think that you'll find a function that'll do all the drawing. And you may have to look multiple times. I see five. A lot of repaints. Okay, so it's called repaint rather than paint. There'll be a function in here that does the painting. There's a mouse dragged. Mouse moved. That one looks like it does some interesting things. Let's move. Repaints. Yeah. What what does the source code above line five forty-three do? There's a bunch of lives I've let's see. Hey, there we are. Line five nineteen and five twenty. That's where we're getting the X and the Y coordinates of that mouse event. Every time it moves, it's gonna call this function and that's how you know where the mouse is. Okay. And so to me, this looks like some sort of a tool tip as you move the mouse over something, it may pop something up on the screen to show you some additional detail in a label. So, it's gonna compare your mouse position to the positions of different objects on the screen. There's some additional information in my. And so I'm guessing that's what below line five twenty-four, that's what that's this code is doing. So, Line right here. E dot get. So, E comes to me like shorthand for. I mean, I'm I don't I don't think there's a class called E. Yeah. So, E, it's highlighted in a gray box and above it in the function definition on line five sixteen, you can see that E is a variable that's provided every time that function's called that is of class mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. So, E is just an instance of a mouse event. There it is. Okay. So, it just returns the value of X. Which is by definition the X position of the mouse. Yes. Yeah, it's really nice in the Java library. They've put all these green comments in there to help you better understand what the code is doing. Okay. Below this, I see some friends we were talking about a few minutes ago. The X depicts and the Y depicts on 525 and 526. So, this is where it's translating the on-screen coordinate system to the physics coordinate system. Oh. Oh. Oh. Oh. That makes sense, Taylor, but it also the nomenclature of the different variables is something I'm just going to have to get better familiar with. So, when we call something X back in that mouse event routine, I assumed that X was already in units of pixels. Yes. But here we have on line 525 something that's calling a routine to convert X to pixels. So, I don't but the I think the usage of the nomenclature X is different in those two cases. Yes. The author here may have reversed the concept in his naming of the X depicts function. Maybe it's picks to X. It would be a better name for that function. And yeah, that's the that's the whole exercise here is we're getting into the mind of this developer to understand how he designed and built all of his code. So, it looks to me like new variables I and J become defined that probably are this alternative coordinate system. So, the X and the Y get translated to I and J. And I and J, I'm I'm coming to understand must be the physically meaningful values. Yeah, rather than pixel values. Yes. So, somewhere down here. Well, probably not in it might be in this one. I and J. Okay. So, how are they going to use that? They're going to look at the difference between I and IC and take the absolute value of that and find out if that's less than. D is defined as a space time object. Stamp this IT dot next. So, the interesting if if the value. So, D is some object that has a certain width on the screen pixels. Yep. It could be like an ST clock or an ST event. Something that's an icon that appears in that highway diagram. Okay. And we're trying to find out if. So, so actually. This makes it appear that the the units of I and J would in fact be pixels. That's what this looks like because it's it's comparing this. Yep. Range of I values to a certain pixel width of an object there. Right. And as long as it's less than that divided by two. That's less than half the pixel width of this object D and as long as the J is also less than half that pixel width. Then we're going to put a first serve. I can't understand the syntax correctly. So, we could, I think we could see this in action if you opened the the space time frame back up again by clicking the Duke icon. And move your mouse over that clock that's at 000. There it changes to a hand icon, a hand cursor. So, by doing that, you're calling that code loop we were just looking through. So, cursor is different from mouse icon. Mm hmm. I think that one's called pointer and if you hover over, it's called hand. And it looks like there's some dragging codes. You could try to drag that clock around. Right. Yeah. Okay. Very cool. Yeah. Alright, well, I need to I think I just need to take some time. You know, the drawing panel seems like a good place to be working just to play with and see if I can figure out what each section is doing. Although, drawing panel is kind of long. 500 lines is a lot of lines. I agree. Might try to find a shorter one for my first one to spend to do a deep dive into and really understand what each block is controlling. Yes. Yes and I think to finish today, maybe I can show you my screen and show you a technique that I use when I begin to analyze software programs. So, that screenshot that I took at the beginning of today's call, I brought into a very simple drawing program and what I've been doing now is as as we've been working, I've been annotating that graphic to indicate what class file is associated with each of the elements on the page. Oh, okay. And sort of building a map knowing that we've now explored the left-hand side of this, what I would do next is go and now go over to the right-hand side and figure out what classes are associated with this table and the different tabs. So, it's probably a class per tab that implements all that. At that point, you'll have figured out at least like at the top level or the user interface identified all the user interface components of the software and then what would be left would be the the animations and then the mathematics. Alright, well, one step at a time. We'll get there eventually. We will. It's a it's a ambitious project but I think it'll it'll do a lot of good once it gets converted to the web. Yeah. Yeah. And I don't think this is gonna be done in one academic year either. I envision this. We're gonna get a good start, whatever that looks like. Happy to spend several more years plowing through this. Myself, I don't know if you have the to to stay with the project for that long and you know, you're you're just generously spending time with me to to help me do this. But this has been the best meeting of my week. I'm interested because I'm in the process of developing a next generation user interface for brain power software and I wanna use the same technology in both applications. So, just like there's swing for Java, there's gonna be a a framework, a library set for the web and it'll have all the same things, frames and mouse listeners, buttons, and we are gonna be building everything using that new framework. And I'm gonna be using the same one to build my new applications as well for work. So, my parting thought is that I think it would be good for us to ask the permission of the original author if we could create a new open source GitHub repository. Because we're gonna wanna use that repository to begin to share code edits. We can check in, we can push, we can pull, I can send you stuff, you can send me stuff, we can, this this documentation we're beginning to create, we can have a safe repository for it. But I'd like to have the author's permission before we post his code to the web. Yeah, sure. Okay. Alright, I'll send him a message and I'll CC you if you would like me to introduce you to to the so he knows who I'm working with. Perfect. Yeah, I'd love the introduction. It's it's always great to meet new people. In Slovakia, nice. He is a physicist over there. Very cool. Alright. Well, thanks so much, Taylor. You're very welcome, John. Thanks for your time and maybe we can do this again next Friday or in a couple of weeks. Yeah, let I will follow up with you. I need to check my calendar cuz there's gonna be some beginning of year meetings coming up. I need to get to the right. Yep. Sounds good. Alright, my friend. Have a good weekend. Take care. You too. Bye. Bye.