Programming Methodology-Lecture20

Instructor (Mehran Sahami): All right. Let's go ahead and get started. A couple quick announcements. [Audio cuts in and out]

Sorry. A little technical difficulty. So when last we left off – hopefully you can hear me – the idea is to create the coolest thing you possibly can while using graphics, okay? So there's two categories that we're actually going to judge these on. One is esthetics, which is just the nicest looking thing. It can still involve interaction, and probably should involve interaction or animation, but just the coolest looking thing. Then the coolest thing allorhythmically, depending on what kind of stuff you put in there. Is it a game? It doesn't have to be a game. Whatever it might be.

So to give you an appropriate incentive for this, what we're going to do is when you turn them in, Ben and I are going to look through them. We're going to take a first pass and take a sub-set of them that we think are the best. Then we'll have the section leaders in the class collectively vote as to what the winners are in both of the categories. You get, at most, one entry. Your entry, you don't have to designate for a category. We'll look at every entry as being entered in both categories. You just get one entry.

If you want to focus on a particular category, you can, but we'll look at both of them. Then we'll have the section leaders vote and see who the winners are in the two categories. Just to make your life a little bit more interesting, your prize, if you're a winner, is 100 percent on whatever thing in the class ends up being your lowest score. It could be the midterm, could be one of the programs, or it could be the final exam. Whatever ends up dragging your score down the most.

So at this point, if you're a computer scientist, which means you're thinking ahead, you could think to yourself and say, hey, if I know I win the contest, because you're going to announce the contest winners on the last day of class, and I know my lowest score is going to get replaced by 100 percent, were I to not take the final exam, would that not be the lowest score in the class and would get replaced by 100 percent? Yes, in fact, that would be the case, which means if you win the contest – if you choose. If you still want to take the final, that's cool, and if it turns out that something else is lower, you can drop that lower thing just as an added bonus if you want, but you don't have to take the final exam. You just get 100 percent on it if you want.

Now, for those of you who are thinking, hey, doesn’t that blow out the curve and cause all these other bad things to happen, no, as a matter of fact, it's goodness for you, too. The reason why it's good for you is whoever wins the contest, we're not going to count whatever that score was that we replaced with 100 percent as part of the curve. So it's a very tiny thing. In a class of 300 people, it really doesn't make a difference, but even for you, the person who won the contest, probably was going to do pretty well on the final anyway or the midterm or whatever. So taking them out of the curve is actually a little bonus to you as well.

The other thing, just to sort of say, if you enter and you're not one of the winners, we don't want to say, all your work was for nothing, immediately. What we're going to do is say, of all the people who entered the contest who have a serious entry – a serious entry means you really put some reasonable amount of effort into it. You didn't just say, here's my hangman program. I'm just going to turn that in as a contest entry. We want to see some real effort. But for all entries that show some real effort into the contest, what you'll get it we'll put you into a random drawing, and on that last day, when we announce the winners in the two categories, we'll also do a random drawing of everyone who actually entered the contest for one additional winner. That randomly drawn person will also get 100 percent on whatever their lowest score was in the class.

So just a little incentive for you. If you don't have time, you're like, oh, I'm so swamped with all this other stuff. That's fine. You can take the little slip of paper about the contest, put it in the recycle bin and just put it out of your head. It won't affect your grade at all. But if you want to go for it, you're welcome to go for it. So any questions about that?

All right. So a couple other announcements. One thing that was kind of interesting was I read all the evaluations, so the mid-quarter evaluations. Hopefully you already did for [inaudible]. If you haven't you should go do them online. But I actually do read them all, and I'm very serious about trying to improve things. So I was looking through, and there were some that were kind of interesting that I wanted to share with you.

One of them was, when I review the lectures online, I noticed you say, Do, do, do, a lot. I had no idea I did that, but do, do, do, now I do. Better throwing accuracy was a pretty common theme. There were some things that I'm not sure how to deal with, so I'll give you these two comments in a row.

He could go a little slower, which I can fully appreciate, and then, perhaps he could go a little faster. I look at how many comments I get on each side to try to weigh that, but there's a lot of comment of that form. There was one that I saw that made me go, hmm. It was, I have never been to lecture. I was like, thanks for sharing, and you probably won't actually ever see this because you don't go to lecture. One person asked – actually, a couple people, I was amazed they asked for more baby or toddler pictures. That one's easier fixed. When I get the overhead camera. There's life in the Sohami household. Notice the book he's reading.

After a little while, you kind of go through the book, and when he gets later in the book, he realizes this whole time, he hasn't actually been using JAVA 5.0. He gets a little upset, but that's life in the city. There you go. There's more toddler pictures. Then the last one, which I thought was interesting, out of all 300 that came in, but I saw Professor Sohami – you can call me Maron. It's fine –at [inaudible] buying a hard drive yesterday. I wanted to say hi, but I forgot how to pronounce his name. I was so embarrassed. It's Maron, and thanks for sharing. All right.

So with that said – and then there were the other comments that I actually pay attention to, not that I don't pay attention to these. If you saw me buying a hard drive, hey, yo also

works. I'm happy to chat with you about the latest hard drive configurations. Anyway, the topic for today's class is our friend, the GUI. And you look at that, and you're like, GUI, what's that all about?

This is what we refer to as a Graphical User Interface. But the way it's usually pronounced, graphical – graphica. It's sort of more avant-garde that way. At times, you'll hear people say this as a GUI. Like, oh, it's GUI. It's like taffy. They're just pronouncing the letters, GUI.

The basic idea in a GUI, right – and actually, if you think about it so far, you've been doing some stuff that involved the graphical interface before where you might've had a program, and you did something that involved mouse clicks. So when the user clicked on the mouse or they moved the mouse, like in breakdown, you were essentially creating an interface for them that was graphical. But the notion of graphical interfaces that most people are familiar with when they talk about GUIs are these things that you interact with on the screen. Things like buttons that you may press or what we refer to as sliders, which are little things that look kind of like this, and you move this button back and forth along some scale from high to low. Check boxes, a lot of times, you see on forums on the web. It's just a little box. Sometimes you can put a check mark in it, or you can click it again, and it takes the check mark.

Something referred to as radio buttons, and this should hopefully be familiar because this was on your midterm exam, except we had you do this very stylized version of it with the graphics library. Radio buttons are just basically where you have a list of choices with buttons, and when you pick one, it becomes selected. When you pick another one, this one becomes unselected, and the other one becomes selected. So if you want to give someone a short-list of options.

There are some other things that come up. For example, something that's called a combo box. This also goes by other names. Sometimes people call it a drop-down box or a chooser. It's one of those things that kind of looks like this. Sometimes you see a little triangle next to it, and it might have some value in it like blue. You click on the triangle, and you kind of get this thing that drops down, and you get other choices like black or red or green or whatever the case may be. That's called a combo box or a drop-down box because when you click on it, this thing kind of drops down.

Or just our friend the text box. A text box is just where there's some form you fill out. You can enter some text in it. Okay? These are, collectively, what we refer to as interactors because what they are, are things that allow the user to interact with your application and provide some information to your application interactively, by clicking on buttons, moving around sliders, setting check boxes or selecting from radio buttons or picking some option in the combo box. That's all they are, so we refer to them collectively as interactors.

Now, the interesting thing is, how do we actually use these in the context of a JAVA program to allow someone to interact with our program more than they've done before?

So far, people have been able to enter text in a console window, or they've been able to move a mouse or click. We want to be able to do something more. So in order to do something more, we need to have the use of some libraries. The libraries that you need to use for interactions, besides the standard ACM libraries that you use, like ACM.program.star, is you want to input two JAVA libraries. One is called JAVA.awt.event.star. These are all in the book, but you can write them down if you want.

You also want to import something that looks a little funkier. It's JAVA X, so just keep that in mind. It's not JAVA. It's JAVAX.swing.star. And collectively, what these do is these help you keep track of events when the user's interacting with something. Before, when you had mouse events when the use clicked, and JAVA X swing is actually a package that's part of the standard JAVA libraries that has a bunch of stuff that allows you to create these graphical interactors very easily on the screen.

So those are the libraries that you're going to have. So to give you and idea of what's actually going to be going on in your program, if we can go to the slides, let me show you how the program or the interactor hierarchy looks. So in the JAVA world – this is in JAVAX.swing. All interactors, in some sense, at the end of the day, are what we refer to as J-components. J-component is kind of the basic, generic thing that all interactors are. It's kind of like when you think about the graphics library. All the elements that are in the graphics library like a G-oval or a G-rect or a G-label, are all G-objects at the end of the day.

Same kind of way to think about it with interactors. At the end of the day, all the interactors are J-components. Notice this is a J instead of a G because we refer to these as – they're really JAVA objects, in some sense, so they all start with a J. That's just a naming scheme in JAVA. Then there's various kinds of things we can have like a J-button is a J-component. It's just going be a particular button that we're going to display on the screen. There's different kind of buttons. There's your standard button.

There's other buttons like a toggle button, which we're not going to talk about because no one ever uses toggle buttons in their raw form. But, for example, a check box and a radio button, if you think about them, in some sense, are ways of doing some kind of selection like a button. A button, you just click one thing. A check box, you set some box to either be checked or unchecked, and a radio button, you pick on of many options. So in some sense, it's just how much flexibility you have with them, but at the end of the day, you're doing some interaction with something that involves clicking somewhere to turn something on or off, basically.

There's a slider, which we talked about, a little slider thing that basically is some spectrum that you can move some controller on. There's something called a J-label, not to be confused with a G-label, but a J-label is very similar. It's just a piece of text that you can put next to some of these other components to label what it is. It doesn't actually do anything other than just sitting there, being a pretty label.

Combo box that we talked about, and a text field, which is basically like a text box. Now, there's a few other things that are in here that kind of come up like entfield and doublefield. We won't be talking about those. Most of the things in here, we're actually going to cover today, and it's just important to see how they're related. They're all components at the end of the day, much the same way that when we did graphics, all the individual graphical kinds of objects were G-objects.

Now, with these interactors, how do we actually put them on the screen? We don't just put them anywhere on the screen. They actually have a special way that they get laid out on the screen. It turns out, now, again, you're old enough to find out something that we sort of didn't tell you about this whole time, even though it existed this whole time. Now it's time for you to know about it.

Your program window, whether it's a console program or a graphics program, actually has five regions on it, okay? So far, you never used any of the four regions around the side. You always just used the center region, but there was actually five regions labeled, sort of by the points of the compass, north, south, east, west and then the center.

So the way this actually works is the center is where all the action was taking place in your programs before. So when you had a console program, what you really got was a text console that took up the center region, which was basically the whole screen at that time. Anything you wrote out, it got written into the console. On the flip side, if you had a graphics program, what a graphic's program did – remember, we talked about a G-canvas. What it did was it put a G-canvas in the center and sort of made it big enough to take up the whole screen. So that's what was going on this whole time.

You might say, but Maron, what happened to these other regions around the side? I didn't see any space getting taken up by the regions. It turned out that the other regions are only visible and only take up any space at all if you add interactors to them, which means if they had buttons on them or sliders or combo boxes or whatever, when you put them on, when you're going to put those interactors on, you're going to say which one of these regions, north, south, east, west or center. Most of the time, you won't put them in center because the action will still be going on in the center.

When you put them in one of the regions around the sides, it says, hey, I have some interactor in the southern region. I need to now show the southern region. It will actually take up space on your screen. Any questions about that?

I'll show you and example of this in just a second. So that's the basic idea of window regions and what's actually going on with them. The one thing that is important to keep in mind, just in terms of name, is when we place interactors in one of these regions, we refer to that region as a control bar. So if we put some buttons, let's say, in the southern region, what you'll see when your program runs is you'll actually get sort of this gray bar down at the bottom, and your buttons will show up on it. We refer to that gray bar with the buttons on it as just a control bar. If we want to be specific, the southern control bar. That's just the name for it.

With that said, let's actually create our first interactor. It's time to actually make one of these things and put it to use and see what it all looks like. Then we'll build something super cool and complicated with them. But let's just start with the most basic one right now. A little side point. Computer science career panel next week. Go there, Wednesday, November 14th in Packard room 101. It will show you a wide spectrum of things that you can do in computer science at 5:30.

There will be people there who are working at start-up companies, people there who went into academia, people there who are, for example, doing product management or marketing who all are graduates of Stanford's computer science program. It just shows you the wide breadth of stuff that's going on. So if you have any inkling at all that maybe computer science is for you, check that out. That's just a little side point in the middle of lecture.

So let's create our first interactor. The button. Okay. The way we create this is we're gonna need to have a constructor. We're going to need to create something called a J-button. So the type is J-button. We give it a name. I'll just call it but for our button because we have one. So what I'm going to do is I'm going to create a new one of these things in the standard style I use for creating something new. It's a new J-button, and what I give the button when I create it is I give it a single parameter, which is the name or the string which I want displayed on that button.

So let's say I want this button to have the word, hi, on it. Okay? What that does, it creates a button object that the button is labeled with the word hi. This button does not yet show up on the screen. It doesn't yet do anything. All I've done is create this button object. Now, the next thing I want to think about when I want to do anything involving these interactors, right? After I create one of these, I'm going to put it somewhere. Before I put it somewhere, one thing I'm going to keep in mind is I need to listen for these interactors.

So before, when we talked about the mouse, we added mouse listeners. We talked about keyboard events, we talked about adding keyboards listeners. If I want to do something with interactors, I need to add what's called an action listener. So somewhere in my program, say in my innate function or my innate method or my run method, what I'm going to do is say, add action listeners. Much in the same vein I would say, add mouse listeners. As a matter of fact, I could have both. Most of the times, in programs that are interactive, I probably will have both.

But add action listeners, when I add that, it says, oh, yoo-hoo. I'm going to be listening for things happening, like when someone clicks this button. So I'll show you in just a second, there's a corresponding function that gets called – or method that gets called when some action offense happens, and that's the idea. When you had a mouse, and your mouse moved or you had a mouse click, you got mouse events, and so you created methods that were called automatically for you when those mouse events happened. Like, when the mouse got clicked, one of your methods might've gotten called.

Here it's going to be a special method that's gonna get called every time some particular action happens, and I'll show you that in just a second. So we need to add action listeners at the very beginning of our program, and then somewhere, we're going to create a button. Then we need to add the button to one of our five regions. So the way we do this is we use add, and we're going to use a version of add with two parameters. What we're going to add is the object, so we specify the name of the object. This should look real similar to you from graphics when you added, say, a G-rect to the window. But because these are interactors, we don't just add them to the window. We add them to a particular region. So we might say South, so capital north, south, east and west are all predefined constants for you that let you know what region you're referring to.

So add a button to south says, put this button in the southern region. Now that there's an interactor on the southern region, it will automatically show up. So any questions about the basics?

Student: [Inaudible].

Instructor (Mehran Sahami): Ah, good question. How does it know where in the region they show up? Well, what it does is it will automatically manage the layout for you. So what it does is any interactors that are in a region who up centered in that region, and they show up in the order in which they were added to the region. So if you add three things to the region, one, two and three, they will show up in the order, one, two and three, and they will be centered, with respect to the entire bar. I'll show you what that looks like in just a second. Let me do a complete program. Let's write a complete program together on the board, and then we'll see what that code actually does when we run it.

So we're going to have some innate method. A lot of times, in interactive programs, rather than having a run to begin execution, you just have an innate [inaudible]. I'm going to set things up, and then I'm going to wait for interactive events to happen. When they do, other methods of mine will get called. So I'm not actually running something. I'm just kind of initializing the world. So oftentimes, you'll see these programs written with an innate method rather than a run method. That's just the way they are.

So what I'm going to do, is I'm just going to do a shorthand. I'm going to do the construction and the add all in one line so you see that. Add new J-button with the label hi to the south region. Okay? So all I've done is I don't have a variable to keep track of this J-button anymore. I just kind of create it and add it, just like you may have done with some graphical objects when you created them and added them immediately to the canvas. We're doing the same thing here because we don't need some method or variable to keep referring to this button. That button, when it's pressed, is going to invoke some other method for us automatically, as long as we add our action listeners.

So then here, we would say Add action listeners. So what we generally do in our innate method is we create all the interactors that we want to create, and before we're done, we

say, add action listeners, to say, okay. I created everything. Now go ahead and start listening for events on them.

Now, when some action actually gets performed, and you've added action listener, what happens is a method gets called for you called action perform. So this method is public void, and you should always declare it as being public void, and it's called action perform. Lower-case a, upper-case P. Action perform. What this gets is a particular parameter, and the parameter it gets is something called an action event, and I'll just call that E. That's the parameter that it gets automatically. Just like when you had mouse events, when you had your mouse clicked function, for example. Exactly the same idea going on here, except this gets called when, for example, someone clicks one of your buttons.

Now, how do you figure out what actually happened when this interaction was actually performed? Action performed, and important thing to keep in mind, only gets called when buttons get clicked. So later on, when we talk about some other things, I'll show you, when someone clicks a check box or whatever, how we deal with that. But the important thing to think about is action performed only called buttons get clicked. So right now, we only have one button, so you might just say, hey, if I click that one button, this gets called, I know it's that one button. Life is good, right?

Well, sort of. You still want to keep track of which button was actually called because you might have an application that has multiple buttons in it. So how do you do that? What you do is you're going to have a string that we'll call CMD for command, which is going to be, essentially, the command that caused this action to be performed to be called. What does that mean? What it means is this action event, E, we're going to pass it a method called get action command. Get action command returns to you a string that we're just going to assign to this variable command.

So we say, hey, an action was performed. Get for me some string that refers to what action was performed. It says, okay. The way I keep track of this command, what is this referring to, this will have the same name as the name of whatever button was pressed. So if a button has the label Hi on the screen, it shows up with the label Hi, ad if it gets pressed, get action will return Hi. It will essentially return whatever label was on the button that go pressed.

So I can check to see if command was equals Hi because I know that's my button, Hi, because I coded Hi up there. I could've actually made this some constant, a string constant, if I wanted to keep them in mind. If it was equal, then maybe I want to do something, like I want to print some lines of the screen.

Printlin, and so this printlin is still going to print to the center region because that's still where the console is in a console program, and I might write out hello when someone actually clicks the Hi button.

So there is no run method here, right? I just set up the state of the world with a button. I start listening for what's going on with that button, and any time that button gets click and action perform gets called, I ask this event that gets passed in, hey, what command was actually issued? I get that as a string, and then I can check to see what button it actually was. So let me show you a couple examples of this in terms of what that actually looks like when it's run.

Okay. So we can quit out of here. Oh, don't save. It's never fun to save. All right. So here's a program, which is basically the program we just wrote. I called the first button. It extends the console program. It includes the libraries that I care about in addition to the ACM library. Other than the fact I set the font to be larger, I just add this new button, Hi, to the southern region, add action listeners, and I have this event here, or I have this method, action perform, which is basically exactly the code I just wrote up on the board.

So when I run this, what I actually get is – here's first button. I get the southern region, which now shows up as a control bar. It has my single button, Hi, on it, and nothing's going on in my program, right? It just creates the state of the world. It puts the button. It says, now I'm listening. Now, every time I click on the Hi button, it executes the command.

After it goes off the screen, it keeps scrolling. You're like, how small can I make this roll button – the scroll bar over here on the side. It's the things you do at 4:00 a.m. when you get your programs working that just make it that much fun.

So let's look at a slightly more complicated program. Same idea, but something more complicated. This is called Button Press, and all Button Press is doing is it's creating three buttons instead of one button, ad it's adding them in different places. So it's creating a button that says hello in the northern region, creating a button that says CS106A in the southern region, and creating another button that's called Basket Weaving 101, also in the southern region.

After it creates all those buttons, it adds them, right. You always remember to add the buttons. It adds action listeners to listen for those buttons. So what happens if a button gets clicked? Action perform is going to get called. Here, I get the action command, and depending on what the action command was, I know which button was clicked. So if it equals hello, I know the hello button was clicked, and I write out hello there.

If it equals CS106A, I write out CS106A rocks, and if it's basket weaving 101, I write out not so much. So here I run my Button Press program, and notice now, the northern and southern regions now show up. So the regions automatically show up any time there's an interactor there. You don't have to tell the region to show up. It knows to show up if you put an interactor there. If I click hello, I get hello there. 106A rocks, and basket weaving 101, not so much.

You're just like, oh, click a little here, click a little there. A little under the arm. Yeah. There you go. It's ten lines of code, and I can put buttons all over and just click them

wherever I want. So buttons are kind of fun, but we can do some cooler things. Let's actually create and interactive program and bring back – remember Click for Face? Do you remember that from a long time ago where we click the button, and we got a face on the screen?

Does anyone remember? If you remember that, raise your hand. Yeah, click for face. This is supped up click for face. The basic idea is we're going to allow someone to click for faces on the screen, but they can pick the size of their face using radio buttons, either small, medium or large. They can pick the color of the face, black, blue, green or red. We'll start with black, and they can choose to either show the front of the face or the back of the face by saying, hey show the front or don't show the front.

So let me show you an example of this. So I'm going to draw a medium-sized face, the front of the face, in black. So I click here. Oh, front of the face, medium-sized. Front of the face, medium size. Large face. Large face. Small face. Small face. Red face, red face. You're like, okay, that's fun, but what does the back of the face look like?

It's the back of the head. The guy's bald. There's nothing. It's an oval, but that's the thing we want to keep track of, right? Do we draw a face or draw an oval, depending on if front is clicked or not. So how do we get all these buttons and these check boxes, these radio buttons, this drop-down box or combo box. And then we have clear, which is just a button that erases everything. Then we're like, yay, large face. And we just start all over again.

So how can we create this whole program in the span of one lecture? So here we go. We're going to go do it. All right? So first thing we're going to do, it's speed. It's sort of like name that tune, but it's sort of name that interactor. It's like, how many interactors can you go through in half an hour? It turns out they're actually not that complicated, so we can go through a whole bunch of them.

So you already saw a button. What are some other things that we want to do? Check box, right? So check box, you saw that check box, that's allowed us to either show the front of the face if it was checked or the back of the face if it was not checked. How do we create a check box?

Well, the type here is called J-check box. For a J-check box, what I need is a variable of that type. I'll call it check. The way I create this is I say new J-check box. The parameter I give the check box is the name that I want displayed next to the check box like front, to indicate is that the front of the face. So it will right out front and put a little box after it.

Now other things I can do with the check box, which are kind of cool, I can set its initial state, to either be checked or unchecked. So what I can do is I can say, check dot set selected and give it either a true or false, a [inaudible] to say do you start off checked or not checked.

So if set select it to be true, it starts of checked. If I set it to be false, it starts off unchecked. Then when I create this thing and set its initial state, I need to add it to some region to show up on a control bar. So I can say, hey, add check to the south region. Now I've gotten a check on the south region. So the thing you should be asking is, hey, now you've gotten a check box on the south region. How do you actually know when someone clicks the check box? Do you get this action performed method called for you when someone clicks the box?

Actually, you don't. It's a little bit different, how you actually check for the check box. It turns out that for the check box, you can check its state, whether or not its checked or unchecked at any point in your program. The way you do that is as long as you have a way of referring to the variable check, you can just say check dot is selected. What the will give you back is a boulion. So we might have some boulion T equals check is selected.

The tells you any time you call that, whenever you call it in your program, is this thing checked or not? So if you think about this, in order to be able to refer to check, what that means is when you actually declare this puppy, this thing is probably not a local variable. This thing is probably actually an instance variable.

So what that means is somewhere in your program, when you're doing innate, say this is your innate method over here. And then public void innate. You would say check equals new check box, but really, somewhere down here where you declare your instance variables, you would say something like private J check box check, which means I'm going to have this instance variables. There's something I need to be able to keep track of in between method calls.

So I'm going to declare it as an instance variable. I will create a new on in innate, but this one is actually just setting this instance variable. So somewhere else in my program, I can refer to that instance variable.

Student: [Inaudible]

Instructor (Mehran Sahami): Yeah, you can actually do the creation there and say J-check box check equals new J-check box and then whatever the name of it is. Oftentimes, you just don't see that, so I'm following that convention here, but sometimes you'll actually see someone do that in a program, yeah.

Student: [Inaudible].

Instructor (Mehran Sahami): The reason, really, for doing it this way is that in innate, you can sort of see how everything's being created. You don't need to look at two different places in the program. There's some things where it actually makes sense to create them when you declare them and other things that you can't create them when you declare them. It doesn't make sense to do that, and it's better to just create them all in one place, basically, is the general thinking.

So when check is selected, this is something that you might check somewhere in your program. For example, when the user clicks the mouse somewhere, right? So when we click somewhere and get a face, at that time, we want to see, are we showing the front of the face or the back of the face? So in our mouse-clicked method, that is getting called automatically when we click the mouse, that's when we're actually checking to see if check is selected.

I'll show you the code in just a little while, but that's an example of how it might actually be done. So instance variable's the important thing to keep in mind there. So besides the check box, we had this radio button for the size being small, medium or large. Radio buttons get a little bit more involved, but not a whole bunch.

So the first thing we're going to do is we're going to create all the individual radio buttons. So I might have small equals new – actually, let me abbreviate these just so it's easier to keep track of them. We'll have SM, which is small, is a new J-radio button, and it's going to have some text associated with it, which is just the word small. So this actually creates a single button, right, the single circle with the word small next to it.

This is not – yeah might say, okay, if it does that, how do I know that it's related to medium and large and all that? We'll get to that. Small, notice I haven't declared a variable out here. Again, a radio button is something that we want to be able to keep track of over time. So small, we would actually declare over here. Let me just increase the amount of space we have for variable declarations by getting rid of innate for right now.

We would declare a small over here by saying private small J-radio – backwards. Private J-radio button small. Okay. So that's going to be another private variable for us. We're going to create it in our innate method somewhere, or perhaps another method that gets called for innate.

We create the small radio button. We can create the medium radio button by doing the same thing, new J-radio button, we'll just put the label, medium, on it, and we can do the same thing for large. New J-radio button, large. So we create the three radio buttons.

Once we create the three radio buttons, we need to say, hey, all these radio buttons are in a group. They're all related to each other because if you click small, medium and large better turn off. If you click medium and small is on, small better turn off. So how do I know that they're all related to each other?

In a program, you may actually have multiple radio button groups, and you don't want them to conflict with each other. So what you do is you create something called a button group. A button group is just another type, and I'll call this size because size is small, medium or large. So our group is going to keep track of size. Equals new button group.

Now, the interesting thing about button groups is I never actually care about referring to the button group after I create it and assign all my buttons to it, that I'll show you in just a

second. So this button group, oftentimes, is actually a local variable. It's not an instance variable because I don't need to keep track of it after I create the original button group.

I need to keep track of the buttons, so medium and large would also be instance variables, just like small, but I don't need to keep track of the group. So what do I do? How do I put all these guys in a group? I say the name of the group size, dot add, and I add all the buttons to it. I would add small, and I would say size dot add and add medium, and I would say size dot add, and I would add large.

That says, hey, now you're group has these three buttons in it. Once I do that, still not done. Oh, let me just slide this over here. The power, the magic, of boards. So once I create my initial group, I want to say one of my radio buttons is selected to begin with. So I pick the one I want selected, like medium, and say set selected to be true.

I only set selected after I've added all my buttons to the group. So the general idea is I create the buttons. I create the group. I add all the buttons to the group, and then I pick which one is selected because it doesn't make sense to select one until I know everything that's actually in the group.

After I do that, the funky thing is I need to add my radio buttons to the control, one of the control bars, but I don't add the whole groups at once, as strange as that may be. I add all the individual buttons. So I need to say add small, south. Add medium, south. You could actually put them in different regions if you want, but then that's going to mess up the user real bad. And large in south.

Usually it's a good idea if they come right after each other, so they're all sequential. Otherwise, if you have some interactor in the middle of them, the user's going to get very confused. So this basically sets up my buttons. I create the buttons. I create the group. I put them in a group. I pick which one out of the group is initially selected, and then I add all the buttons from the group onto the control.

Now, how do I figure out which one of these buttons is actually selected somewhere in my program. It's because I made these things private. Small, medium and large should all be private instance variables. I can refer to them later in the program by saying something like small dot is selected. This gives me back a boulion, which tells me, out of that group of radio buttons, is small the selected one?

So if small is the selected one, the other two are probably going to be false. But I can check for all three. I can say medium is selected and large is selected, same kind of thing. Anywhere in my program, it will just give me back the appropriate state of whichever radio button is selected at that given time. Again, it's not calling the method, that action performed method only gets called when buttons are pressed.

Things like J-radio button is not a standard button, so it doesn't call that method or check box doesn't call that method either. Any questions so far? All right.

So one last thing we need to have before I can show you a program that puts it all together, and the last thing – anybody remember what the last thing that was that we showed in our little sample program for the face? Combo box. Always a good idea to have a combo box.

I'm not actually going to show you the slider. The slider is pretty straight-forward. It's all in the book. It's not worth spending lecture time on it, and the truth is, when was the last time you saw a slider on a web page. You're like, today. Yeah. Yeah, that's for today. That's for tomorrow, and that's for the rest of your life.

Yeah, slider you can read about – oh, the snag. Denied. That was one of the other comments, too. The social candies come too far in front, and I just can't go that far back. There's only – I've been hearing, oh, you should really practice your – a lot of the comments were, you should really practice your throws. I've been doing this for 15 years. It doesn't get any better, okay?

All right. So we want to have the combo box, okay? The combox. I always want to write combox, but it's combo box. They should just rename it the combox because really, you can just reuse these two letters and reduce global warming. So the combo box. How does the combo box work?

We have some variable called the J-combo box. We'll call it pick for our pick for the color. We're going to create a new J-combo box. Once again, combo box is something that we want to keep track of between method calls. So we would not actually declare the variable here. We would refer to the variable, but we would declare it over here in our private instance variables and have private J-combo box pick. Okay?

So we start off by initializing the pick to be an empty combo box. This just creates a new combo box. Then what we need to say is we're going to add all the elements that we want in that combo box in the order in which they're going to appear in the drop-down menu of the combo box. We do that by saying pick dot add item.

Then the item you give it is a string. We would have black for the first item, and then the next item pick add item is blue. I just put these all in alphabetical order, but you can put them in whatever order you want. Green, and then pick add item red. Those are all of the choices that are in the combo box. The reason why it's called a combo box, interestingly enough. You might say, but isn't it just a choice box or drop-down box? Why is it called a combo box?

The reason why it's called a combo box is in the days of yore when these things were first created, they not only let you pick from a choice of items, but you could actually type in a value if it allowed you to, and just make up your own value that wasn't in the list. But oftentimes, we don't want people making up their own value. If we said, hey, from this combo box, pick your year.

You could pick freshman, junior – we don't have sophomores. Freshmen, sophomores, juniors, seniors, grad students or other. If we let you just type something in, you're like, well, I'm supersenior. First of all, I'd be like, sorry, but second of all, I'd be like supersenior doesn't exists. You're still just a senior. You could be super senior, but you're not supersenior, all one word.

So what we want to do is tell this J-combo box, hey, we're not going to allow the user to type anything in. We're just going to allow them to pick one of the choices. The way we do that is there's a property called set editable, that we're going to set to false. So we say set editable false. If we set editable to true, you will get a combo box that allows someone to pick any one of the choices or just click on it and type something in, if you really want to do that.

But if you don't know how to handle what they type in, don't let them do it, and set editable to false. The other thing that we need to do is we need to pick an initial value for that combo box. All right. So you create a combo box. Which one of the choices is initially picked? We do that just like you've seen for all the other interactors. Pick dot set selected. Except here we say selected item, for some reason. With combo box, we just use the word item when we add and set selected item because they're just items. That's the way life is.

Item, and then we give it the string of whichever one is selected, like black will make our initial choice. After we do all this, we create the combo box. We add all the choices to it. We choose whether or not it's editable, and we pick the initial selection. Always remember to say, hey, now I need to add it to one of my control areas on the screen.

So I say add pick to, for example, the south controller. Okay. Question?

Student: [Inaudible].

Instructor (Mehran Sahami): I think the default, that's actually – offhand, I don't know if the default is false or true. I think it might actually be true. But just for finding something that I don't know off the top of my head, there's a candy. And for asking a question. You can just try it out yourself. So don't set it and see what happens. Offhand, I don't remember. We could actually just play with it in the program.

So here's how all those things actually get put together in our program for drawing this interactive face. So first of all, what I'm going to do is in my initialization method, I'm going to create the buttons that I care about.

I only had one button, which was the clear button. So I have one button, clear, that's in the southern region. What do I do when I get the clear button? Remember, buttons are the only thing that the action performed method is called for you automatically when they're clicked. So down here, I have my action performed method. It's pretty similar to what you saw before with the hi button except this one gets an action event. It gets the action command. So here, I didn't take this and assign it to a separate string.

I'm just saying, hey, E, get your action command. I know that's going to be a string, so just directly call the equals method on it. Oftentimes, you'll see the shorthand when there's only one thing that we want to check for instead of a bunch. We don't get the string separately.

So E action command equals clear, which means they click the clear button. If that's the case, then I remove all. Remove all is one of the methods of a graphics program that removes everything that's on the canvas, so it clears the canvas. So that's how my button gets set up. Then I have a check box for whether or not I'm displaying the front or the back of the face. So what do I do? I create a new check box that's name is front. Notice I need to keep track of the state of the check box, right, because I need to know later on, was it checked or not.

So check box is actually down here as an instance variable. We use our instance variables to keep track of the interactors whose state we need to refer to as the program runs. We need to say, hey, check box, are you checked or not, somewhere later in the program. So we need to be able to refer to check box.

My radio buttons are all also instance variables, and my combo box pick, just like you saw up there, right? Radio buttons, combo box, and the check box were all instance variables. Same thing going on here.

So check box, I create it. I set it selected to be true. So when I first start off, I'm going to be drawing faces, and I add the check box to the southern region. Now there was two other things that I had. I had a radio button for small, medium and large, and I had a color chooser, which was a combo box. [Inaudible] to take all of that code and abstract it a little bit and modularize it, I've just created two methods that I'm going to call to do that initialization for me.

So innate radio button. What that does is it does the code you just saw. It comes here, and it says, hey, create a button group. Create three buttons for small, medium and large, and add all the buttons to the button group. Same code you saw. Actually, you could've created the button group. You could've taken this line of code and done it after you created the buttons. It doesn't make a difference, but here, I just created the group, created three buttons, then added all my buttons to my button group here.

Once all my buttons are in there, I say, hey, medium is the one that starts off selected, and then add all of my radio buttons to the southern controller. Any questions about that? It's exactly the same code we just wrote on the board, except just with nicer variable names. Uh huh?

Student: [Inaudible].

Instructor (Mehran Sahami): Right. They all get drawn from left to right, but the entire collection of stuff is centered in the screen. So you'll see that when we run the program. It's not going to happen.

So last but not least, I initialize my color chooser, and my color chooser is just a combo box. So I create the combo box. Again, pick color is an instance variable. I add my items to it, black, blue, green and red. I set its editability to be false, so I don't allow the user to type in. I set the selected item, initially, to be black.

Now here's something funky that I'm going to do. I'm saying, hey, here's my combo box. My combo box doesn't have a name associated with it. When this thing actually gets drawn on the screen, what I get is a little drop-down box that looks like – actually, I just get a box that starts with black in it. If I click on this little thingy over here, I get blue and red and green and the other choices.

What I would like to have, is I would like to have the word color, if this chalk actually writes, appear before the combo box so the person knows, yeah, that's what this black is referring to. It's referring to the color of the face. So the way I do that, is I'm just going to create a J-label. I don't worry about assigning it to anything because the J-label doesn't do any interaction. It just puts a label in the control bar, so I create a new J-label.

I add some spaces here to separate it a little bit more from whatever the last control was. So whatever interactor I put in before, I say, hey, before you draw the label, put in some space, then draw the label. So I add that label to the southern region. After the label, I add the combo box.

So what will show up in whatever interactors I had up to this point, then a little bit of space in the label color, and then my combo box after it, okay?

Last but not least, I say, hey, add mouse listeners, and add action listeners. So I set everything up in my program, and I'm done. Now the only thing my program will do is do something when the user interacts with it. So when the user interacts with it, there's two things they can do. They can either action perform, clicking the clear button, which you already saw, or they can click the mouse somewhere on the canvas.

If they click the mouse somewhere on the canvas, we call the mouse clicked method, just like you saw before, clicking the mouse before. Here's where all the fun goes on with checking the state of the check boxes and all that. We say, hey, I'm going to have some generic objects, some G-object.

The diameter of it is going to be whatever size should be appropriately set. So I'm going to call some method set get diameter size. Diameter size comes along and says, hey, I'm going to start off by setting size to be zero, and then I'm going to check which radio button is selected. Hey, small, are you selected? If you're selected, then the size of the small diameter. If you're not selected, then I'll check medium. If medium's selected, then the diameter's the medium size. If it's not, I'll check for large, and I'll set the large size.

Small, medium or large are just constants that are set in my program to be 20, 40 and 60. So all this does is check to see which of the three radio buttons is actually selected and

returns an appropriate value for the corresponding sides. That's what we set for the diameter of the object.

Then we come here, and we say, hey, check box, are you selected? If you are selected, the I want to draw a face. So my object is going to be a new G-face. You're just using that G-face class I wrote two weeks ago. If the check box is not selected, your object's going to be a G-oval because the back of the face is just the circle.

Then lastly, I need to set the color. I'm going to set this object's color, just using the set color method that hopefully, by now, you're very comfortable with, to be whatever color the current color is picked in the combo box. So I write a method called get current color that returns to me a color object.

It goes through, and it says – here's how it works. Hey, color-picker, what's your selected item? So it returns a selected item. I need to cast that to a string. That's just a little JAVA-esque thing. This is all in the book. You need to cast whatever the selected item is to a string, and then based on the string, I say, hey, is the string blue? If it is, I'll return the color blue. If the string is green, I'll return the color green. If it's red, I'll return the color red, and if it's some random thing I don't know or it's black, presumably, I'm just going to return black. So that returns the color object black, which is what I set the object's color to.

Now I've set the size of the object, whether or not it's the front or the back of the face, the color of the object, and now I'm just ready to add that object wherever the mouse was clicked at a [inaudible] location. So that's the whole program. When I run, here's how I get the layout. I added the button first. Then I added the check box. Then I added the radio buttons in this border. Then I said, hey, a little bit of space, and the label color, and then here's my combo box for this lecture.

Select green, set this to be large, I have front. I get the green face. Any questions about interactors? All righty. If that's making sense, nod your head. If it's not making sense, shake your head. All right. Have a good weekend. I will see you on Monday.

[End of Audio]

Duration: 51 minutes