Programming Methodology-Lecture05

Instructor (Mehran Sahami): All righty. Let's go ahead and get started. A couple of quick announcements before we start today – so hopefully you're all busy working away on Karel and life is good. Just quick poll – how many people have actually finished Karel already? Oh, yeah. I won't ask how many people have not yet downloaded Eclipse. There are no handouts today. Getting' a little breather – no handouts. Don't worry; you'll get some more of that next time.

Sections start this week, so hopefully you all should have gotten an email about your section assignment and who your section leader is, so you can actually do Assignment No. 1, the email portion. You should have been able to do the programming portion the whole time. But make sure to go to section this week.

And the other thing is the Tresidder Layer, which every once in awhile you've heard me refer to. This is a computer cluster up in Tresidder. Is staffed by one of six helpers like almost continuously around the clock or most of the times at reasonable times when people are working.

So Sunday through Thursday, every day except Friday and Saturday because contrary to popular opinion, computer science people actually do have lives or we actually like to pretend we have lives, but every day from Sunday through Thursday 6:00 p.m. to midnight there will be a staff of helpers on there and actually some of the times there is like two or three or four people there.

And they're there just dedicated for the 106 classes. They're not like general consultants. They're just there to help you work out problems in this class, and they know like what assignments you're working on, the whole deal. They're all like, your section leaders and they're all be exceptionally trained to do this.

The other thing that's going on, hopefully you should be doing Assignment No. 1. I've actually gotten a bunch of Assignment No. 1 emails that have already come in. In the early days, when I got the first few, I actually tried to respond to them all, but then at some point, I just woke up and I like, you know, went to my computer and was like, "Oh, you've got mail.'

And it just [inaudible]. So I couldn't respond to everyone individually. I apologize if I don't respond to you individually, but I do read them all. I guarantee you that I actually read them all and I look at backgrounds. And just to prove to you that I do, here's some interesting ones that have come in so far – just to share three.

So there's someone actually spent their time in Taiwan living in a Buddhist monastery, which I thought was interesting, except for the fact they were actually living there as a monk whom I thought was pretty interesting.

Someone else used a – I wasn't quite sure on this concept, but maybe I can provide a clarification. There was a vegetarian who only eats low-quality meat, and so he mentioned that as things like burgers and not steaks. And I would qualify that by saying that's not a vegetarian; that's called being a grad student.

And last, but not least, there was actually someone in here who's on the Colbert Report, which I thought was actually pretty interesting. I don't know in what context, but come talk to me afterwards.

So with that said, any questions about anything before we start? Today we're actually gonna go over some of the graphic stuff you saw, talk a little bit more about objects and classes and get into variables and values and all kinds of goodies. Any questions?

All righty, then let's just dive right in. So one of the main topics for today is this thing called a variable. And a variable, you know, like variables come up in mathematics and it's like, oh X and Y are variables, right and there are these things and we do all these manipulations on variables.

In the computer science world, they're really friendly, right, and you don't have to worry about integration or differentiation or you know, those kind of variables. Variables are kind of your friend, and basically all the variable is in the computer science worlds is it's a box. It's a box where we stick stuff and the stuff we stick into that box can change.

That's why we call it a variable because it's a box that has a variable contents, and you think back you know, in the days of yore in math, and you know, oh yeah, it's kind of like X can have different values, yeah, it's basically just like that.

So in computer science, what we think of as a variable, is each variable has three things associated with it. It's got some name, and that's just how we refer to that particular box. It has a type, which is something a little bit different in mathematics, but the type basically says what kind of thing does this box store? Some boxes store numbers; some boxes store letters; some boxes will store other things, like little objects in the world. But a type is just what's stored in that box.

And then there's a value, and the value, as you can imagine is just what's in the box. What is the actual thing that's in there, right? If it stores the number, then it might store, for example, the Value 3 and that's just the value, and it may have some name associated with it.

And how do we actually name these? There's actually a rule and it's not a very complicated rule, but a very simple rule you need to remember for what are valid names for variables in Java. So a valid name, so this is how you actually name these puppies, has to start with a letter or an underscore. So it starts with a letter like one of the alphabetic letters and can be upper or lower case or the underscore character.

Okay, and that's kind of you know, underscore. It's down at the bottom of the line, okay? And then after you have that initial letter or underscore, then after that, you can have any number of letters, numbers, that's like the number digits, like 1, 2, 3, 4, you know, 0, etc. or underscores, okay?

So you can't start with a number. You have to start with a letter or underscore, but for most purposes in this class, just thing about them as letters. When not actually using underscores, you might occasionally use numbers. You can actually have numbers after the first letter.

There is one slight caveat to this rule which you can't have any variables that's name is the same as some which known as a reserved word in Java, which means its name can't be the same as some special word in Java, like the word class is a special name in Java, and there's actually a page in your book, I think in Chapter 2, that lists all the special names. It's like out of the English language it has about 127,000 words. I think there is like 40 in Java that you can't make a variable name.

Okay, you have lots of other choices. As a matter of fact, there's lots of things that don't have to be valid words in English. They can just be any name that follows this rule. The important thing to think about in terms of a name and this is one of the good software engineering principles, is make your name descriptive. If you have a program that's maintaining for example, the balance in the bank account, a real good name for the place where you store the value of the balance would be something like "Balance."

A real bad name would be something like "A," because no one knows what A is. It's like hey it's A – yeah, I know what A is. And someone says, "Yeah, A is balance," and someone says, "No, no, no, in my program, A is actually how many miles I bicycle today," and you're like, "No, no, no, man, A was balanced." Well, if it's called Balance, there's just no ambiguity, so give them descriptive names.

So that's kind of a name part of this. The next thing is what is this type all about. What are the different types that you can actually have. And there are some things that we refer to as primitive types. These are the types that have the smaller developed brains and use knuckles drag on the ground. No, there just the types that are built into Java, okay and some of the basic types we have is something we call an INT, which is short for an integer, but we actually write INT, so INT, I-N-T, is the name of the type, okay.

And this is just an integer of value. It's just gonna store some whole number basically. It stores a number between minus 2 billion and plus 2 billion, but for all intents and purposes, you can just imagine you can store any integer in there, okay?

There's also besides integers what other kinds of numeric data do we have? People already know, it's like I would think it would be like real values, but a bunch of people are already saying double because you've read ahead and you've done the assignment the way you should. And that was kind of a mini social, but both went to the same person.

There's this thing called a double and a double is actually some real valued numeric value, right. It's something like 2.3 is a double or even 2.0 can be a double, okay? Why is this thing called a double – anyone know, as opposed to like a real? Uh-huh?

Student:[Inaudible].

Instructor (Mehran Sahami): Yeah, there's this wonderful verb people call the I triple E, which is like the Institute for Electrical and Electronic Engineers and they come up with all these standards for things. Is anyone a member of I Triple E here? No one? Oh, man, join and pay your dues. It's a good time. I'm not actually a member myself.

But what those folks actually do is they come up with standards for things and one of the standards they came up with is how you represent numbers that are real valued numbers inside a computer, right, because remember a computer only understands ones and zeros, so how do you actually represent a real valued number.

And so there's a standard, and part of that standard has to do with a precision of the number, how many digits in some sense and a rough approximation you store and double stands for a double precision real number.

And so for the purpose of this class, all real valued numbers that we're gonna use are just of type double. Okay, there's a couple of other types that I'll just mention now and we'll actually go into them in much more detail in a couple of future classes.

One is called Boolean. And if you've ever heard of Boolean logic, this is a logical value, so this is just essentially true or false, and we'll talk about that in excruciating detail next time, but I'll just let you know that there's a type called Boolean.

There's also a type called care or Car and as you can imagine it's because we like sort of the first syllable of most things and this is the first syllable of character, okay, and so we'll also talk about this in a couple of weeks time when we actually get into a some things with characters, but that's just a character.

It's a variable; it's a box, but still is a character. It's a box that stores an integer, box that stores a real value, box that stores true or false and a box that stores a character. Those would be the different types of them.

So one thing people – so that's kind of types, at least some of the basic types. And then when you think about value that we actually store in this box, people always get uncomfortable when they see INTs and they see double, right, and they sort of say, "But Meron, like 2 is an integer, right?" And I'm like, "Yeah." And they're like, "But you just told me 2.0 is a double, right?" And I'm like, "Yeah, 2.0 is a double."

And so the natural question is why? Why do we have both these things? How come like all integers aren't subsumed by the real values, right, you're kind of the mathematician type and you're like, "Yeah, there's like strictly more of these than there are of these. So

why are all these like subsumed in here. Why do we have this integer type," okay? And the reason we do boils down to a simple question and the simple question you want to ask yourself is how much versus how many, okay?

So if you ask someone, let's say you, ask me, just so I won't embarrass you, "How much do you weigh, Meron?"

Student: How much do you weigh, Meron?"

Instructor (Mehran Sahami): I weigh about, you know, 155.632 pounds, okay? That makes perfect sense, right. If I can put a decimal point, I can put as many numbers after it, or I could just say 156, right and those are both valid kinds of things.

Now you could ask me, "Hey Meron, how much children do you have?" I have 2.3 children. Does that make any sense to you? Yeah, it's like we had three until that grisly accident. No, it's just that there's sometimes – I know that's horrible to say. We actually have one, and he's just fine.

There's sometimes in the world when you care about counting, and when you care about counting, it doesn't make sense to have fractional values. Those things are integers. They're a how many kind of value. When you're thinking about how much, that's a double value, and you actually want to keep these distinct because if I ask you what's the next number after 1, you say –

Student: 2.

Instructor (Mehran Sahami): 2 because you're thinking integer, and that's perfectly right and if I say what's the next number after 1.0, you say –

Student: [Inaudible].

Instructor (Mehran Sahami): Yeah, you start mumbling and you're like, "Well, I know it's not 2.0. That would have been the obvious answer and that's probably not it. How about 1.0000001, and I'm like, "No you missed a couple of zeros in there. Just keep going. Wait until the end of the quarter and put that 1 and come back and talk to me."

In real values, there is no next value, okay, so when you care about counting, it doesn't make sense to use a double. That's when you want to use an integer, so having these things be a distinct type actually make sense and you'll kind of see that as we go along, okay.

So the value is basically just what you're actually gonna put into this box over here and it's gonna be some of the values that we actually talked about, so let me show you some examples of how we might use variables, what the syntax for variables actually is in Java.

So what we need to do with variables before we use them in Java is we need to declare a variable. It's kind of like declaring your love. If you're gonna have some box somewhere that's got something stored in it, you need to come tell the world, "Hey world, I've got this thing called X. It's cool. Check it out." Here's how you do that in Java.

You say INT X, okay? What that does is you specify a type for your variable, what's the box gonna hold. You specify a name for that variable. In this case, the name is X and because all statements end with a semi colon, just like you saw in Karel's world, we put a semi colon.

What does this do? It creates a box somewhere in the computer's memory that's name is X that's gonna store an integer in it. Right now, what does it store?

Student:[Inaudible].

Instructor (Mehran Sahami): Nothing. We haven't told it anything to store in particular. And so in Java's world, unless you initialize you always want to think about initializing a variable, giving it some initial value to start off with and there's some rules that we'll talk about as we go along regarding when Java will automatically sort of give you a zero value in there and there's time when it doesn't and sometimes it can get a little bit confusing.

So the general rule you want to remember and it's also good software engineering is give variables an initial value when it makes sense, okay? So how might we give it an initial value? Well, on that same line, rather than just putting the semi colon there, we could say X equals 3 and what that does is it gives it an initial value of 3 to x.

Okay, so you could actually have the declaration of the variable without the initial value, but you always need to give it an initial value before you use it, so a lot of times we just give it the initial value when we actually declare it, okay.

If we want it to have something of type double, maybe something called Y, we could have something called Double Y equals 5.2 and what that means is somewhere in memory we have this thing called Y, which has value 5.2 in it, and it's just a real value.

So the general form for how you do these declarations is you say type and I'll put little squiggles underneath these just to let you know that this is the general form, the name and then potentially an equal and some initial value, followed by the semi colon, and that's how you do the declaration.

And so the question that comes up where do you actually make these declarations and the place you generally at least for the time being make the declarations of value, for declarations of variables is inside methods.

So when you create some method like you have public, void, run and you're gonna put everything that's gonna go inside run in there, you could, for example, have this line in

there and this would create X with a value of 3 and Y with a value of 5.2 and you wouldn't have the general form there, but for example, you could have those declarations inside run and what that means is you have those variables created with some values available to you inside this method called run.

They're only available to you inside that method called run. They're not available to you in some other method and we'll talk about how you get values passed around between different methods in a few days, but for the time being when you have your variables they're only available in that method. Okay, so any questions about that – variables or values? Uh-huh?

Student: If you have them inside the public run method [inaudible], will that method be able to take that variable?

Instructor (Mehran Sahami):Well, you won't be defining another method inside run, right; you'll be defining another method that's separate from run. It's just a separate method. It won't be available in that separate method. It doesn't matter if it's public or private. It just not available there. So this public or private thing that you have here doesn't affect the visibility of your variables. Your variables are always only visible inside the method in which they're declared and set up, okay?

Now, one other thing that we did when we sort of, you know, did this equals thing over here, if we wanted to, we could have actually done this in two lines rather than setting up X initial value there, we could have said into X and then we could have set X equals 3 over here and gotten the same effect.

And when we say X equals 3, this equals is not equals in mathematics; it's actually what's known as an assignment statement in Java or in most programming languages. And the idea is we're taking some value over here. That's what's on the right-hand side of the equals and assigning it to whatever variable is listed over on the left-hand side of the equals, so we're assigning that value, okay?

And so the general form for an assignment, let's just do that over here, is pretty straightforward, you just saw it, is variable and this is the name of the variable. So I'll squiggle – equals some value or some expression for a value and we'll talk about expressions in just a second with a semi colon at the end of it, and that's how you assign a value.

You can assign values multiple times to a variable, to a variable. It's a variable, the contents change. So you say, "Hey, X equals ," and then down here somewhere, "Hey X equals 4," and when you get to this line up until you get to that line from here to here, X will have had the value 3 and when it gets to this line, it'll stick a 4 in the box. And that's perfectly fine because X is a variable. As long as everything you stick in there is of the type of which X is declared, you're okay, okay?

So what that also means is when you're doing Assignment you can do things that are perfectly fine in programming, but to mathematicians, they just go nuts; they go crazy like veins just burst in your head and you hear these popping noises and people dying in the streets, which is you can say, "Hey, I have some variable called total. Let me create some variable in its total," which I'll give some initial value like 10 and that's a good time.

And then I say, "Hey total equals total plus 1." And if you're a mathematician, you look at that and say, "No, man. Total can't equal total plus 1. That's just not right." And then you got this whole philosophical question about, you know, if this is like total is equal to infinity and you're just like, "No, no, no, that's not what we're talking about."

We're talking about assignment. This is into equals, this is an assignment. So what it says is evaluate the right-hand side, okay? Total – what was total's previous value. Go and look it up in the box over here you have some box for total. Its value was 10. It looks it up. It says, "Oh, that was 10." I'm gonna add 1 to it, that gives me 11. What do I do with that 11? Stick it back in the box named Total. So it just says okay, and it puts an 11 over here and that's perfectly fine.

So it's perfectly fine to do something like then where you take the old value for a variable, you do some manipulation on it, you stick it back in the same variable. Okay? So with that said, now that you know about variables, we can go back to a couple of the programs that you saw last time and things ill now begin to make a little bit more sense, so if you come to the computer, remember our little friend, add two integers that you saw last time.

Suddenly the world opens up a little bit and we say, yeah, add two integers is a console program, right. So it's gonna write some text out to a console somewhere and we have our run method. We print out a line; we know that the method print [inaudible] when we say that, whatever we put inside the double quotes gets printed to the screen so it prints out this program adds two numbers, so we execute that line and after we execute it, it writes it out and it comes to the next line.

Low and behold, what have we done here? We've declared a variable named N1. What's the value for that variable that we're gonna assign to it? We're going to call some method. The method we're gonna call is something called Read INT and this is a method that's provided for you in console programs. All console programs can do this. The way it works is you give it some text between two double quotes. It writes that text on the screen just like you saw last time and asks for input from the user.

Whatever value the user types in and hits enter, that number is the value that gets what we call return by read INT. It's the value that [inaudible]. It's kind of giving it back to you. Do what you want. What are you gonna do with it? You're gonna assign it to the Box N1, so here's the box for N1.

We execute this statement. It's coming; it's gonna print to enter N1 on the screen. It's gonna ask us for a value. Let's say we type in the value 17 and it sticks that 17 so this expression over here evaluates the 17. It sticks 17 in the box for N1, goes to the next line, now it declares N2. We have some box. Its initial value also is gonna come from the user via this read INT method.

So we call read INT. It asks us for another value; we type in 25, and it sticks that in the box. Now, we're gonna declare another; we just declare until the cows come home, right? Declaring is fun; it's cheap. It's easy; do it, do it often, right, so we're gonna declare another variable total, so we have another box over here for total and the value of total that 's gonna get assigned there is just whatever value is in N1 plus N2.

So to evaluate this side, it says I need to look up the box N1, so at the point where it reaches this line, it evaluates whatever is in N1. This isn't some truth that holds for all time. It's just an assignment that it does when it reaches that line. So it says, find out what's in N1, add it to what's in N2.

That'll give you the value 42; stick that in Total, which is what you get. And the last line says, I'm gonna print something to the screen where the total is and you see these funky plusses and you're like, "Whoa, whoa, whoa Meron, I thought plus was light lading things together. What's going on? You're like trying to add – this thing's got the value of 42. I know that because it's in the box. You're trying to add 42 to some text? That's not right."

Well, in most worlds, it would not be right, but it turns out in the Java world it's perfectly fine and when it sees something that some words with a plus sign, this plus sign here is not numerical addition anymore. This is essentially you can think of as a concatenation operator. It says, "Take whatever text is here and add to that text, the textural form of this value."

Well, the textural form of 42 is just the Characters 4 and 2, so it sort of adds that 42 here and then it adds to the end of that a little period, so it concatenates all those things together and then prints them out and says the totals 42.

So any questions about the program? If you're feeling good, like you understand what just went on in the program, nod your head. Rock on. That's what I like to see. All right. So with that said, we can go back to something else we did last time, right, so now you know how all this variables and assignments and types and you're like, "Hey, Meron, yeah, that's all good and well, but last time you were telling me about like classes and objects and drawing stuff on the screen like, what's up with all that."

And this is the time when we actually do sort of the, you know, Reese's Peanut Butter Cup. Like it's two great tastes that taste great together. We take some objects over here and some variables over here and we just slam them together, and now you're going to have variables that contain objects. They taste great, all right?

So here are some of the objects that you considered last time, right? We had things called, like some classes. You have the G label class and the G –rect class and the G-oval class. These are all these graphics classes like a label [inaudible] or rectangle or an oval. I'll show you examples of all these things this time.

And these guys were actually all classes that are part of a hierarchy of other classes and the other classes, they're all part of is G object? So what G-object is a G-object is a class. Don't let the name fool you. Some people see G-object and they think, "Oh, that must be an object." No, G-object is a class. It just means a graphics object. And guess what, a rectangle or an oval or a line or a label or all different graphics objects. So what that actually looks like in terms of the hierarchy just like you saw last time, is the classes are represented, kind of like this in the hierarchy, all of these classes are actually G-objects. Okay, they're all sub classes of G-objects. It means they're specializations. So any behavior that G-object has, all of these things have, but they might have their own specialized behavior, as well.

Now, the interesting thing about all this is that when you have classes and you create objects, you can store those objects in a variable. You can say, "Give me a variable that is a box," and the type that that box holds is an object that's the type of some class, so all classes can actually be used as type means, okay?

Now, that's kind of funky. Let me show you an example, okay? Here is another program you saw last time. Now, we can just kind of reveal the mystery about variables. We have this program, Hello program that extends the graphics program, right, so it's gonna draw us some pictures.

Well what are we actually doing on this first line? Hey, we're doing a declaration and assignments on the first line. What's the type of our variable? It's a G-label. So we use the class name as the type. The name is label, so we have a box called label and what that box is gonna hold is any object of the class G-label. Okay, so that's the type. It's gonna hold any object.

And the question comes up, how do we get an object of type G label. And here's the funky part. The way we get an object of type G label is we have to ask for a new G-label.

So we give the name of the class here and class is depending on the class, you know, see some examples of this will take in or give in some initial values for what we refer to as parameters, and these parameters are all separated by commas. But at some initial values, that this object uses to sort of set itself up. It initializes the objects.

So objects rather than just taking one value, can potentially hold multiple values in them. In this case, the G-label is gonna have Hello World as some text that it's gonna display in some location that's 100,75.

So we say, "Hey G-label, you're the class. I don't have an object of you yet. You're the class." When I say, "Give me a new G label, it's sort of like going to the factor. You sort

of show up at the G-label factory and you say, "Yo, G-label factory. Give me a new object," and G-label factory is sort of sitting up there and says, "Well, what kind of new object do you want? Yeah, I can give you a new G-label, but how is that initial G label going to look," and you're providing these parameters that sort of specify what the initial G-label looks like.

So this line is now created a G-label, okay? And it has some initial, so when we execute that line, it has some initial text that it's gonna display and somewhere it's also storing the value is 175 because it knows where that's gonna be on the screen.

Now, what we're gonna do is we're going to tell that object to do some additional stuff. The way we tell an object to do something is we specify the name of the variable for the object, okay. So the object name is label. That's the name of the variable. And so that's what we say here. We don't say G-label. We say the name of the variable label.

Then we have a dot; then we have the name of the method that we're going to call for that particular object. So it's a little bit different than Karel's world, all right. In Karel's world, every time you call the method, Karel was the one executing that particular method.

Here when you have multiple objects, you need to specify which object you want to execute this method. So you say, "Hey, label, yeah, I'm referring to you, buddy. Set your fonts to be this particular font, san seraph which you saw last time." It just makes it big. It makes it 36-point font. And then you say, "Hey label, yeah, you, same guy, or gal or it or whatever it may be. Set your color to be red."

And it says, "Okay, I'll set my color to be red, and then finally if I want to take this label and stick it up onto my graphics canvas which is actually the program I'm looking at. Right, I've done all this stuff and my program's still bringing out a blank screen. It's like here you go, nothing going on here. I say add and the name of the object that I want to add, right because now I'm holding this little object called label.

This is my little felt Hello World, and I want to say, "Hey take the felt hello world that's so cute and just stick it in the world," so we add it to the graphics program, basically and it shows up in the screen where it knows it's supposed to show up at a 100,75.

So the general form for this, the people will refer to when you see something like this, you have name of the variable and the method name and the Java-esc names that we give to these, just so we hear other programmers refer to them, you know what they're talking about, is they say the object is the receiver. So the object here is the one who's receiving or being told to do something, okay?

So the name of the variable is the receiver and the method that's being called is called the message and so there's this whole notion of sending messages, kind of like you could think of programming as I am, right, and in your little I am you have like your friend, Bob, and your friend, Sally and your friend, Dave, and your friend, Label, right. And

Label is just sitting there in I am and Label is not very talkative and once in awhile you can send a message to I am which is a form of message that label understands and then Label will do something.

So this is called the message that we send to the receiver, is kind of the nomenclature. But a lot of times, you'll just hear people say, I made a method call on the object and it means exactly the same thing. Again, we like to give funky names to simple concepts. Okay, so any questions about that, uh-huh?

Student: If you add the label first, then you start [inaudible].

Instructor (Mehran Sahami): Yeah, you can still, after you sort of put the little object up there that says, "Oh, the object's in the world, and say, "Object change your color to red," and that's where it's a lot cooler than the felt animals you had when you were a kid, because it'll just change to red right there on the screen because once it's been added, it's up on your canvas, and now any messages that you send to it will actually change its appearance on the canvas. Okay.

So with that said, let's kind of go through and understand a little bit more about what are some of these messages or methods that we can send to particular kinds of objects. So to do that, first let's talk about the graphics world that we live in. There's a couple of things you want to think about the graphics world.

How do we specify all these things like locations and how is it laid out. Well the origins, the upper left, so the upper left corner of your graphics window is location 00. Every thing in your graphics window is measured in pixels.

Pixels are the little dots on your screen, so if you get out a magnifying glass and you go up to your monitor and you look real close and you get like eye strain and then you want to sue, don't worry. It's just fine. At least you saw a pixel and it brought you al little closer to Nirvana. So a pixel is just a little squares essentially on your screen that could actually get displayed.

All the numbers that we refer to are in terms of pixels. So when we think about X and Y coordinates, X is how far you go across, sort of across the screen and Y is how far you go down. The X coordinates just increase to the right from 00 and the Y coordinates, different than what you're used to in mathematics, Y coordinates go down. So normally, we think of this as a negative Y direction. Not so in Java's world. This is the positive Y direction going down.

Hey that's just the way life is. It's been like that since I was a wee tike, or actually, since Java came around. Okay, and so you might wonder, "Hey, if I have some label, like Hello World, and I specify some location, where is that really on this Hello World thing and so the G-label coordinates, the coordinates that 100,75 that we gave it were the X and Y location of the base line of the first character. So that first pixel right there on the lower

left hand corner of the H, okay? And so that's 175. That's how things were laid out. Uh-huh

Student: Is it possibly to specify [inaudible] using ratios?

Instructor (Mehran Sahami):Well, everything's in terms of pixels. There's some stuff you'll see later on in the class where we actually specify some things like in polar coordinates if you're familiar with polar coordinates, you'll get to that later on.

But you want to think of these numbers as just being absolute pixels and you can do some math to figure out where those pixels should actually be. But they're still just pixels, just way off.

All right, so remember all of these classes are organized in some hierarchy and everything that we talked about, right, G-label, G rect, G-oval and G-line are all G-objects, which means any behavior that a G-object has, all of these puppies have. So let's start off by thinking about what are the behaviors or methods that are available on G-objects.

So one is set color; you just saw an example of that, right, you specified the object, you say set color and you give it some color and I'll show you where the colors come from and it sets the color of the specified object to the color you just specified as this thing called a parameter. So the thing that's inside the parentheses in our formal speak, we refer to as parameter. So methods have parameters. They're the values that you provide inside the parentheses when you invoke a particular method, okay?

So there's a single parameter there, color. There's also something there called set location, which takes an X and Y coordinates. Again X and Y are in pixels and it sets the location of that object to that XY, so this method has two parameters, X and Y. And you might say, "But Meron, when I created the G-label before, didn't I specify the X and Y location when I told the factory to give me one of these G-label objects?"

Yeah, you did, but you didn't have to, and it turns out there's times you don't want to; you want to just say, "Hey give me a label that's got the words Hello World in it and later on I'll figure out where it's actually gonna go on the screen, so I'll specify what the X and Y are," but until you specify the X and Y, it doesn't know, but you can specify after you've created the object, is the important thing.

And move, and here's the funky thing. You can actually tell an object, like you have your little, you know, Hello World somewhere on the screen and it's kind of like the furniture idea of objects. You're like, "Yeah, I don't like how it looks there. Move it a little to the right. And so there's a move method and its coordinates if you're sort of a Calculus person or DX and DY, and if you're not a Calculus person, don't worry, you don't need to be a Calculus. This is the closest we get to Calculus in this class. Rock on.

DX and DY just means how many pixels in the X direction and the Y direction should you move this object from its previous location and these can be positive or negative, because you can actually have something negative say in the Y direction and it'll move it up on the screen.

So it's just the offset or how much you want to, so think of it as the difference in X and the difference in Y, how much do you want to change where it moved. Uh-huh, question?

Student: Do you use a set location XY and then [inaudible]?

Instructor (Mehran Sahami):Can you do set location and so you –

Student:[Inaudible] later define X and Y.

Instructor (Mehran Sahami):No, if you want to say set location XY, you're referring to two variables X and Y, so there's variables X and Y and each have already been declared and have values.

Student:[Inaudible].

Instructor (Mehran Sahami):Yeah, so anything that's a parameter here, you can actually use a variable rather than an actual value, like an actual number, and we'll see some examples of that as we go along. [Inaudible]. So where do all these colors come from? It turns out these colors come from a standard library that someone in the Java world wrote that's called the Java AWT package, so if you want to use these colors at the top of the program, and I'll show you and example, that should say, importJava.AWT.star and these are all the colors, so they're all the name color, dot and then what the color actually is.

So you just saw an example where we used colored dot red to make Hello World red, but there's all these different colors and they're all in your book, so you don't need to scribble them down hurriedly, but you know, different shades or black or grey or white and you know, magenta and sienna if you're sort of a color photography sort of person, but there's a whole set of colors and you can just go to town on them. Okay.

So all G objects, all of these things respond to these methods because these guys are all objects, so any of those three methods will work on any of these objects, of objects of any of these types. But there are some additional behaviors, there are some additional methods that some of the specialized classes, the sub classes actually implement.

So G-label has some additional things that it does beyond just what a G-Object in general would do. And the big one, well first of all, it's got what we refer to as the constructor. You can think of the constructor and I'm gonna do a little violence to the term, but you can think of the constructor as a factory, okay? What the constructor says is, "Hey, I want a new one of these guys," so I use the word new and then I specify the constructor, okay?

The constructor is the name of the class of which you're going to get a new object and then some initial parameters. So the initial perimeters for a G-label for example, could be what you just saw, the text that you wanted to display and its initial X and Y coordinate, okay.

There's some other ones. For example, you can change the fonts, right, it makes sense for a label. It doesn't make sense for a square or rectangle, to say, "Hey, square, you're font's gonna be Helvetica." And the square is like, "I'm four lines, man, I don't have a font. What are you talking about?" That's why the notion of setting a font isn't something that all G-objects have. It's only something that G-labels have which means the G-rect and G-oval and G-line don't have a set font.

This is just something that you can do with a G-label and what it does it says specify the font that you want that label to show up in on the screen, and the general form for specifying the font – this is all in your book and it shows you examples of the fonts, but the general form is inside double quotes, because you're gonna give it as a piece of text, you specify the family of the font.

That would be something like times or Helvetica. The style of the font, which is, you know, plain or bold or italic or bold or italic, so yeah, now in fact you can do all those funky things your word processor does and then the size of your font, how big or small you actually want it to be and you specify that as some texts and that would be what you send over to this sent font method to set the text.

Okay, so any questions about that? Uh-huh?

Student:[Inaudible].

Instructor (Mehran Sahami):No, you get them all through graphics program. Rock on. All right, so how do we draw some geometrical objects? Have another question?

Student:[Inaudible].

Instructor (Mehran Sahami):If you're doing a graphics program without any colors, you don't need Java.awt.

Student: But if we're doing one with color, then we need Java.awt in addition?

Instructor (Mehran Sahami):Yeah, in addition to a graphics program, and I'll give you an example of that in just a second. I have to show you a program that has it.

So drawing geometrical objects – it turns out some of these other guys have funky things that you can do specifically with them. So first of all, there's the constructors, the factories from which you can build new things. So how do I make a new rectangle? I say, "Hey, I want a new G-rect." I specify the X and Y location. It's the upper left-hand corner of that rectangle and then I specify the width and height of that rectangle in pixels.

Okay. So that's the basic idea is upper left-hand corner and then width and height of the rectangle.

Similarly, for oval – ovals are kind of funky because you look at this and you're like, "Hey I have an X and Y and I have a height. I didn't like –width and height like ovals. I thought an oval was defined by like two [inaudible] and that whole thing where you have a string. Did you ever do that – you stick the two like nails in the board and you put the string around it and you draw anyone? There's like two people – yeah, like sorry.

Do it; go get a piece of wood, stick two nails in it, put some string around it and go get a pencil, something you can draw ovals and you can draw a 1 and if you're like 6 years old, you'll draw like 1,000, and if you're 18, and draw one, you're like, "Yeah, that was just stupid."

But the basic idea behind an oval is the reason why we specify this way is you specify sort of an imaginary rectangle for the oval and the oval shows up as an oval that just happens to touch the sides of the rectangle, which is how we sort of specify how wide and broad it is.

So think imaginary rectangle and sort of the four ends of the oval would sort of be touching it, and I'll show you that momentarily. G line is very straightforward. G-line, the way we specify line just like you know, the days of yore with Euclid, line is defined by two points, right, that's an infinite line, but we can think of where it ends at the two points. So we specify an initial XY location and an ending XY location and it will just draw a line between those two points, pretty straightforward. Uh-huh?

Student:[Inaudible] is the set location defined as the lower left corner of your object, and the G-rectangle [inaudible] left corner?

Instructor (Mehran Sahami): Well, for set location is the lower left corner of for textural objects, and then it becomes different for rectangles and ovals and stuff like that, yeah. So it's a slight variation because that's just because we're trying to deal with, you know, texts and rectangles symmetrically and it's tough to do, okay.

There's also some additional methods that are just shared by G rect and G-oval. They don't actually apply to lines or to labels for that sense, rectangles and ovals can be filled, so they can either be an outline. You can either have a rectangle that looks just like this which is an outline or it can be filled in which means it's just a big solid chunk of stuff. So you said if it's filled, you say this is either true or false with the actual word true or false.

So if you've set it to true, then when you draw it on there, you say, "Hey, put this thing up on my canvas." By adding it to the canvas, it will show up filled in; it will show up solid. And if this is set to false, it just shows up as the outline of the object, whether or not that's an oval or a rectangle.

And almost done here. There's also a notion of a set fill color, and you might be wondering, you're like, "But Meron, you told me that set color. Here's where things get a little funky, you told me that set color is something that was defined for a G-object, so if set color is defined over here, doesn't that mean that a G-rect and a G oval already can set that color?"

Yes, that's true. "So what's the set fill color all about?" And this is a slight variation. When you set color for something, you set the color for the whole thing if that's solid or if it's an outline, you're setting the color for the whole thing.

When you set fill color, you are just setting the color for this internal portion. So imagine like this was your rectangle, what color used to fill it in can actually be different than what color the line is. So if you want the line, say, to be black and you want the fill to be red, you can actually do that by setting the fill color to be red and then the outline is still whatever color the actual outline was if you want to do that.

Okay, so if the color – if the interior is fill, so the set fill has to be true, then you can set the fill color to be different. That's a minor thing, but it's kind of fun if you're drawing stuff, okay? So any questions about that? Oh, wait – we have a question over there, uh-huh.

Student:[Inaudible].

Instructor (Mehran Sahami):Oh, in the place of this word fill?

Student: Yeah.

Instructor (Mehran Sahami):It's either the word true or the word false. So if you say true, set filled is true which means it will be a solid thing; it will be filled in. Uh-huh?

Student: If your when you're using the [inaudible], you specify a [inaudible], what is the why would you use the set location [inaudible].

Instructor (Mehran Sahami):Maybe you wanna do some animation. Say like you have like your bunny in the world and he's gonna want to move along.

Student:[Inaudible].

Instructor (Mehran Sahami):Well, it's set somewhere to begin with and then later on, guess what you're actually gonna be doing a program that involves animation. Rock on. And you will move things around. So that's where it comes from. All righty.

So one last thing that might be useful to you, sometimes you want to say, center something in the graphics window or draw something in the bottom, how do you know how big the graphics window actually is.

There's two methods from the graphics program class, so these methods you might notice they don't have any objects, that is their receiver, and the reason for that is that both of these methods, their receiver is the graphics program, so if you don't specify the receiver for a method, the thing that is actually receiving the method is the encapsulating class.

Okay, just like in Karel, when you had Karel and you said move and you didn't specify an object for the move, the move is going into Karel, so get width and get height. We'll return to you the width and the height of the graphics window in pixels.

So these actually give you back a numeric value that you can, for example, assign to a variable and then do manipulations on it. Uh huh.

Student: If you don't fill in [inaudible] something like set color to change the entire background of the graphics [inaudible]?

Instructor (Mehran Sahami): Well, you can make the huge rectangle the size of the graphics window and set it to be filled and set it to be a color and then you can change the whole background that way. Uh-huh?

Student: Can you then change the size of the graphics window?

Instructor (Mehran Sahami): There is some stuff for later on, but for right now, just think of a fixed size graphics window. Yeah, so let me push on just a little bit if we can because what I want to actually show you now is a graphics window that makes, or a graphics program that makes use or a bunch of this stuff so over here, here's a little program called fun graphics, okay.

And so we have a comment up at the top and here are all the imports you need. Notice I [inaudible] for you. We need acm.graphicstar because we're gonna write a graphics program. We need acm.program.star because all programs that you write in this class are gonna need that. And because we're gonna use colors we additionally need Java.awt.star. If we weren't gonna use colors, we wouldn't need that, but it's needed for the colors. Uh-huh.

Student:[Inaudible].

Instructor (Mehran Sahami): It did; it might have just been hidden because sometimes it just appears hidden until you expand it out, but it actually had it in there. So here's what we're gonna do. Let me just run this program, show you what it displays on the screen and then show you how we actually get that display, so it's just like, "Hey Meron, you're just drawing garbage on my screen," no, there's really like a method to this madness. Not really, there are multiple methods to this madness [inaudible].

I know, sometimes computer scientists should not be – that was actually not a canned joke. That just came tome – how horrible is that? So here is what's going on. We drew a bunch of stuff on that screen, how did it get there. Well, hey Hello World; we probably

knew how that got there. This should all be familiar to you now. We got a new G-label. We had some variable called label that we assigned it in. We set its font. We set its color to red. Because we set its color to red, that's why we need www.Java.awt because we're using colors.

We set its color to red and we said, "Hey, I got Hello World, throw it up on the screen," so we add label. You give it the name of the variable and there it appears.

Over here we say, "Hey G-rect. I want a new rectangle that's upper left-hand coordinate is 10,10 and it's size is gonna be 50,50, which means it is a square. It's not filled in. I haven't specified its color right," so if you don't say set fill, it's not automatically filled in. If you don't specify the color to automatically black and then I add it and then you get that little box all the way up in the upper left-hand corner.

Then, I say, "Hey, you know what, I want some other G-rect because it's cooler." It's gonna have colors and it's gonna be bigger and better and it's gonna be nationwide, and so what it's gonna do is I'm gonna have rect 2 which is another variable of Type G-rect.

It's perfectly fine for me to have multiple objects of the same type which they knew G-rect that's upper left-hand coordinates is 300,75 and whose size is honking – it's 200 by 100, so it's big long thing, and it's gonna be filled in and I'm gonna set its fill color to be red, so the whole thing is red, throw it up there and so what I get is that big, rectangle. Just ignore the big oval one in front of it for now; I get this big red rectangle up there.

And I say, "Well, that's kind of cool, but I want to see what this oval thing is all about. So Hey, Meron, make me an oval." And I'm like, "All right, well, G-oval will make you an oval," and the dimensions of the oval, its upper left-hand coordinate and its size are exactly the same as the rectangle, and the reason for doing that is to show you the oval in relation to the rectangle that you specified.

It's set filled is true and it's set fill color, not its set color, but its set fill color is set to be green, which means the internal dividend will be green, the outline of the oval will still be black, because we did not set its color and then we added and there you get that green oval and you can notice its four ends touch the same four ends as this rectangle and if you look real closely there's actually a green line that demarcates the oval and then the middle fill is actually green.

It might be a little hard to see, but that's the case. Then we also have a line, so I will just call this my funky line, and my funky line is a line that starts at 100,150 and it extends to the location 200,175 and there is my funky line. Yeah, it's a funky line.

All right, and then I add another thing and you're like, "Dude where's my line." And so Dude where is my line is a line that should cut across the entire graphics window because it starts at 00, well not the entire graphics window. It starts at 00 in the upper left-hand corner and goes to 100,100, and this is a common error which is why I how it to you.

You look up there and you're like, "But Meron, there's only one line," and that's your funky line or my funky line as the case may be. Where is Dude where is my line? And it's not up there. Why is it not up there?

Student: [Inaudible].

Instructor (Mehran Sahami): Yeah. You did not add it to the graphics content, okay. Common error – people will go ahead and create something and add all the colors and set all the sides and be like, "Ah, I'm rocking," and then they'll run their program and nothing shows up and they start tearing their hair out and beating their section leader and it's a bad time, and they're sitting there contemplating life in jail and then they realize, "I just should have added it to the canvas, all right."

But you can't give that as an excuse to the judge. All right. So any questions about that? All right. Let me very quickly – I need to push on real quickly. So I want to tell you a little bit about expressions before we finish up and we'll do more expressions, but you need to know some basic stuff on expressions.

So all an expression is is you already saw one. And expression is something like INT total, equals N1 plus N2. This is an expression. It's just some mathematical expression that tells us how things kind of relate to each other.

And basically all an expression is it's a bunch of terms and these terms can be either individual variables or they can be constant values, like they ca be like the Number 2 or 3 or they could be some method called like Read INT that actually gets a value from the user, for example, and there's some operators between them, like the addition operator, and so when you take some terms and you put operators between them to combine the terms that's called an expression, all right.

Fairly straightforward stuff – you've probably seen these. There's a bunch of operators you should know about. The operators are things like plus, minus, times, which is actually a star or asterisk, which is how we refer to times. It's not as X – division and then strangely enough, this thing's that looks like percent and it's not the percentage operator, although you might think that. This is what's known as the remainder operator, okay.

So these puppies generally work like you would think, add two numbers, subtract two numbers. There's also a uniary minus. The uniary minus is the funky mathematical way of saying negative sign. So you could actually say like equals negative N1 plus N2. That's like a negative minus. That's perfectly fine. Okay, or you can use that same thing, you know, as subtraction if it's between two terms.

Multiplication, division, the thing about division that is funky is it depends on how division is used, okay and I'll tell you a little bit more about that next time, but I want to talk about remainder right now.

And remember back in second grade, when you did like short division and you said, "Oh, I had 5 and 2 – how many times does 2 go into 5 – 2 and you were like remainder 1. Anyone remember that? Wasn't that such good time? I loved remainder – like that's the online like math. I'm like – and when the remainder was 0, I got pissed.

But that's what remainder was all about. It's just what's left after you do the division. So that's the way you should think about percentage. It's the remainder operator, and you're like, "So how does that work?" So what that means is if I say 7 remainder 2, the value of this is 1 because it's after I divide 7 by 2 and I see how many whole times 2 goes into 7, what's the remainder, it's 1, okay? You could say something funky, like, hey what's 7 remainder 20? You're like well, 20 doesn't go into 7, Meron. So that would be 0 remainder – rock on.

So that's the way it works. It's not the modulus operator. If you're a mathematician, or if you've worked with other languages and you've seen clusters and you're like, "Oh, that's the modular operator."

It's not in Java; it's remainder an dif you try to apply it to negative numbers, unexpected things, what's not unexpected; it's just not what you would have expected in the mathematical sense. So as far as you see it in the book and everything we do in this class, just apply it to positive integers and you can only apply it to integers so there's no remainder with valued doubles, there's no remainder.

And so when we come back next time we'll talk a little bit about some of these expressions. Any questions before we break? All right, and I'll see you – oh, there's a question. Why don't you just come up and ask it. Thank you.

[End of Audio]

Duration: 52 minutes