Programming Methodology-Lecture28

Instructor (Mehran Sahami): All righty. So welcome back to our last fun-filled, exciting class of CS106A. It's always a little sad at the end of the quarter, at least for me. For you, it might be a lot more exciting that you're done with this class. But so we have a bunch of announcements. There are some things to get through today, namely some things also related to your final exam, as well as some unfinished business with the graphics contest.

So there's two handouts today, the last two handouts except for the final exam which are a practice final and the solutions for the practice final. The problems that are on that are actually taken from – most of them. Some of them were written just for that, but most of them were taken from actual final exams in the past. So just like the midterm, it should give you a chance to get a notion of what kind of questions we would ask, what sort of topical coverage there would be, the level of difficulty, and I've left the blank pages out of the exam just to save trees, but we would have space in the exam for you to actually take in.

And we give you the solutions so you can check your exams as well. And again, I would strongly encourage you to do this as a way of studying for the final exam, just to get a sense for how comfortable you feel with the material because there is kind of a range of stuff. As I mentioned today, today is the last class lecture. There is no – to sort of pay homage to what's supposed to be dead week, although most classes don't really have a dead week, our attempt at dead week is a dead day, so there is no lecture on Friday, although sections are still happening this week, so go to your section this week. There are section problems, and there is a bit of a review happening as well, so sections are still very useful to go to this week. Assignment No. 7 you know is due next time. You only need electronic submission for that because we're not gonna do interactive grading, so there's no interactive grading for Assignment 7, but it is due at what would be class time on Friday. You just need to submit electronically. Don't worry about paper version. Just wondering. Anyone – how many more people – or I should say how many people are done with Assignment No. 7 already? Just wondering. Wow. A fair number of folks, so hopefully it wasn't too excruciatingly painful.

Course evaluations, there's actually two sets of evaluations, and I would ask you to please do them. They're actually really important. The two course evaluations there are is one is a university-wide course evaluation. And actually, if you do your course evals for the class that you're in, you get access to your grades earlier in Axess, so sort of provides a strong incentive to actually do the course evaluations. But I would encourage you to do them, and be honest in the course evaluation for whatever you think because that's information that we can also use to try to improve the class in the future, so it's useful for us in the long term, and the university actually uses it for a bunch of stuff for it, so it's very important for them. The university, since they have their own eval – we have a specialized eval that we also do. Sort of like you took the mid-quarter eval that was online, we have an end of quarter evaluation for 106A that also gives us some very specific feedback about this class: how much time you spent on assignments, what assignments you liked, what assignments you didn't like, stuff like that – that we actually – It's totally anonymous, but we use it for some data analysis to kinda figure out how are we pacing the class, is the class actually achieving the objectives that we're trying to get to, and you'll find out about that second evaluation in your sections this week. And that one's also very important, so I would encourage you to go to section, find out about that evaluation, and take it. But the other one's the university-wide one.

A couple other announcements – the final exam as I mentioned before, but I will mention it again – it is an open book, open note exam. Bring printouts of your other programs if you want to. You can bring the text book for the class. You can bring your Karel reader if you want, although I can tell you right now that Karel will not be on the final exam. And I'll give you a list of topics as to what's actually fair game for the final, and what kind of stuff is covered in the book that's not gonna be on the final just as part of our final review. But you can bring the Karel course reader if it just makes you feel better to look at Karel. You're not gonna need to worry – or maybe there's some algorithm in there that just reminds you of how to write Java. You're welcome to bring it, but you're not actually gonna need to know Karel for the final. It is closed computer, though, just like the midterm exam, so if you have a laptop, don't bring it along, or if you bring it along, don't turn it on.

The regular exam is Thursday of finals week, December 13th. That's just over a week from today, 12:15 to 3:15 p.m. in Kresge auditorium, and these dates, and times, and places are all given to you on the front of the practice final anyway, so you have them as a reference. And the alternate final is the day before, Wednesday, December 12th, 3:30 to 6:00 p.m. in Kresge auditorium. You are welcome to take either exam. You don't need to e-mail me. You don't need to make prior arrangements. All you need to do actually in terms of making prior arrangements, you need to figure out which exam you wanna show up for, so the only prior arrangements you need to make are with yourself. Pick which exam you wanna go to, and go to it, and it's all good. It just provides you with a little bit of flexibility if you wanna go home a little bit earlier or if you had a conflicting class because I know some people are watching the class on TV. SBD students, I'll say it one more time, and it's the last time I'll say it because it's our last class. E-mail me by 5:00 p.m. today if you plan on taking the exam at your site. Hopefully, I've already heard from all of you that this applies to, but if not, you have your last chance for like another hour after class today.

And last but not least, anyone find a cell phone in here? It's not mine, but there are someone who's in this class who unfortunately has lost their cell phone, so if you do happen to find a cell phone in this class, just come and bring it up to me after class, and I will make arrangements with the student to get it back to them. So any questions about any of the administrative kind of stuff before we sort of delve into some unfinished business? So we have a little bit of unfinished business in this class, and the unfinished business is the graphics contest – well, besides our final review. And I've gotta say that the graphics contest – last time, I told you they were just impressive. I think they were more than just impressive. Some of them were just jaw dropping, so I'm gonna show you actually, and announce the winners of the graphics contest, as well as show you the winning entries. And we'll also do the random drawing to see of all the people who entered who gets the bonus prize.

So our first winner, and if we have a little fanfare, I can kinda slide this to the side. Actually, I'll slide both these to the side. So we have two categories. We have algorithmic and aesthetic. In the algorithms category – algorithms, I can't even spell these days anymore. Is David Tobin here? David, come on down. So good job. As part of winning, there's a couple things you get. One is you get a bag of candy, a whole bag, well, not the whole thing. You get one of these. You can pick which one you want, though, so rummage through. A Milky Way fan?

Student: [Inaudible] Milky Way.

Instructor (Mehran Sahami): All right, that's always good. And we'll show you – we'll demo your program and show what's actually going on, so the program that David did was a program to do fractals. And so I will show you that program, and what is particularly cool about the program. So if we do fractal graphics, it kinda starts off, and it shows you a fractal.

And you're like oh, that's kinda cool. You need to do some math. You need to figure out some stuff with fractals. But as many of you know, fractals you can kind of zoom into at any range, so we can pick a range over here, and it will automatically zoom in and expand out, and we can just keep zooming into the fractal if we wanna keep going further. Now besides just that, that's kinda cool in itself, there's also some built-in fractals. So there's a few that David has lovingly provided for us that we can kind of look at, and that's kind of cool in itself. Now the thing that really made me go, "Wow, that is really cool," is not only are there these built-in fractals, there's a little text box down here. And in that text box, you can type in an equation for a fractal that includes exponents, and you type in the equation, it actually figures out – it parses your equation, runs the math on it, and then generates the fractal.

And you can do this – it's sort of freeform. I was kinda playing with this other day, minus X raised to the fourth, so this X plus X squared minus X to the fourth down here, and it takes a little while because it's rendering every pixel on the screen how it should look. And so you can get these really cool effects, and then we can always zoom down. So it takes a little while to redraw, but it goes ahead and redraws it for you. So not only is it doing all the graphics, but it also is providing the opportunity for you to just type in an equation for it to understand the equation and render it, so very nice work. Thanks very much. And so as you know, for that effort, you also get 100 percent on anything in the class, which can include the final exam, which means at this point you can take your practice final and be like recycle bin if you choose to take the hundred on the final, which means you just don't show up, and when I see that little glaring gap I know oh yeah, that was 100. That doesn't apply for everyone else, okay. Also, in the aesthetics category, is Sally Hudson here? Come on down. It takes a little while. You also get your choice of the random candy. And you might wonder why there's random candy because well, I just buy a lot of candy at Safeway. Kit Kat fan? Good times. And what Sally did for her program – this is also pretty – well, a lot of the programs actually have very good algorithmic and aesthetic qualities, so it's hard to just pick one for the other, but it's a program called the Tessellator.

And I'll show you what the Tessellator does. It starts with a little square. Anyone a fan of M. C. Escher? A tessellation is just where you put a bunch of little things on the screen, but they all have to fit together. So you can click on any part of this square, and to guarantee that it'll tessellate, the part of it that should allow for the tessellation to happen, basically sort of the corresponding piece on the other side, also kind of expands out. We don't want the tessellation to overwrite itself, so we can just kind of pick random areas, and that creates little places – or we can go this way – that create the tessellation. You might say, "That's kinda cool," but now we click set tile to set what the tile is. We've now defined what the tessellation tile is. We can change its size. So let's say I make it a little bit smaller because I like small tessellations, and that's kinda cool. But now I can pick multiple tiles of multiple colors, so let's say I want three different colors. Let's see. I have little my red, blue, green sliders over here, and notice the color changes as I slide, so I slide over here, so that's kind of a puce. I don't know. Well, we'll set the color. So it indicates your color over there. There's color No. 1. Let's just go – as the big Stanford Cardinal fans, we'll just go for red, although cardinal is slightly off from red, and I don't know where it's actually, so maybe it's got a little blue in it? Maybe a little green, too? Let's just say that's cardinal, although you would look at that and say, "But Mehran, that's really salmon." That's cool, too.

And then we'll just pick another color where we'll go, "We won't go white." That's where you have all the – you could even just teach red, green, and blue with these three sliders by themselves. What happens if we put red and blue together? Oh, that's too close to the other one. Let's just go for some blue, so we set the color. Now at this point we go tessellate. And you'll notice because of the way the piece is drawn, you can tessellate everything together of the multiple colors. And you might think that's kind of cool, and that's kind of cool in itself, but it's kinda even cooler if the tessellating pieces decide to pop out and bounce around, multiple of them at the same time. So very nice piece of work. Thank you very much. And this keeps going until they're all done, and you can actually just reset and do the whole thing all over again, and that's pretty cool. Now both of those programs are pretty cool, and you both get hundreds for your work. There was one program that was entered that when I showed it to the – well, first when I saw it, I went, "Oh my god," because it made you weep in a really good way. And then I showed it to the section leaders. Ben and I were demo'ing these because the section leaders all picked the winning entries, and they saw it. And as we went more and more into the different pieces, every time we showed a different piece, everyone just went, "Ah," and it was just unbelievable. So Chris Miel, are you here? Come on down. This is an additional award we're giving called the peoples' choice award because the entries were so good that we figured why stop at two. Because when it comes down to it, in this class there is some level of stuff that I want you to know, right? And if you demonstrate that you know that stuff, we can have some more awards. It's not a problem. So Chris, first of all you get to pick a bag of candy.

And now I will show you Chris' program which is Zelda. All right? So it begins with this, and you're like, "Oh, that's very unassuming." So you start with the beginning graphics screen. If you look at this one thing, you kind of look at this for a while, and then you realize there's a little flash of light that goes across. Someone had to animate that, and they had to think about it. And so you can load a game, or you can start a new game, so I'll show you a new game. It has some nice graphics in there where again if you actually read everything – anyone in here Zelda fans? Yeah, I was when I was much younger. It's actually a very funny sort of take on the Zelda concept. Notice, if I turn different ways he's – oh, getting a little feedback there. There's a shield that I can use in different directions, and there are different screens I can go into, like I can go into this house over here, and here I can buy stuff. Like I've – I'm a little bit here so my hearts are down here, so I can say, "Hey. I wanna buy a couple more hearts," and I'll buy a potion, too. And here's my funds over here. Here's my magic potions and my emeralds that I buy stuff with. And I'll exit.

And then I just kinda cruise along, and there's stuff that goes along in the world including things that like fade. Notice the fade effect there. And as you continue to cruise along in the world, you see various kinds of things that maybe we wanna kill, and just kind of cruise along. And you get to places where there's just animation going on everywhere, like the water's animated. I'm getting beat by the octopus. Never let yourself get beat by an octopus. And you can't do things like run into the river. It actually is aware of what – aw, I died. That laugh goes on a little too long. There's a whole world you can explore. You can save the games and come back. I should be using my shield, but I'm kinda ignoramus with the shield. Oh, yeah. That's right, buddy. Sometimes when you kill something, it turns into an object. Sometimes it doesn't. And it was like every time you enter a different screen, it was like this. It was just astounding. So – oh man, I haven't even seen this screen – [inaudible]. I know what I'm gonna be doing over winter break, so I will save you all the rest of the time.

We'll just go in here, so you can see some stuff that's going on. Music changes. We probably can't actually show this on the video because I think are actually sampled from the real game, perhaps? Yeah, that's live from the city. But thank you. That's just an astounding piece of work. So sort of what I refer to as the peoples' choice award goes to Chris Miel. Am I pronouncing that right? Is that Miel?

Student: Yeah, it's Miel.

Instructor (Mehran Sahami): Miel. All right. Good times. The other thing that came up is these, you get 100 percent on anything which is nontrivial, right? You're getting 100 percent on the final, and I looked at this, and I said, "I could give Chris 100 on the final, but I think he's pretty much demonstrated anything that I could possibly have wanted him to know in this class, so I'll just give him an A plus." So you're done. That's your grade. [Inaudible]. Thanks. And if any of the contest winners if at any point in your career you should need any recommendations, let me know. I'll be happy to write it. Now there were all sorts of honorable mentions. So these are all the winners of the contest. There were some honorable mentions, and I wanna recognize a couple of the honorable mentions that really stood out. Unfortunately, we don't have time to demo them all. There were a lot of cool programs, but a couple honorable mentions as well. Jasmine Mann? Yeah, you wanna come on down? And also Paris Georgegoudis? Am I pronouncing that right? Are you here? Oh, just too good to come to class, but we'll sort of give it to him anyway, Paris G. So Jasmine, you are also entitled to candy.

Student: Yay.

Instructor (Mehran Sahami): And so one of the things you also get, as well as Paris who's not here, just for the very strong effort that you put in, is you know there's a participation grade in the class. So you automatically get 100 percent on your participation grade. So thanks very much for a nice piece of work. Jasmine did a musical piano that displayed some stars and had music, and Paris actually did a backgammon program with a computer player as well. So now there's finally a little more unfinished business to do which is that besides all the winners, there is also a random drawing for everyone who put a serious entry into the contest, and both Jasmine and Paris are both eligible for that potential 100 on the final. If you get the 100 on the final, though, you don't get the 100 on the participation because you only get one prize. It's your choice because if you happen to win, you get that 100 on whatever you want, so you can still use it for participation. But in terms of everyone else who's in the contest besides the winners, there was 14 total people who were in the contest. So I figured why don't we just pick the winner collectively in a random drawing, and so I wrote a little program, which just pales in comparison to Zelda. Actually, I even took this program from someone and modified it. That's how low budget this effort was because it was just a cool program. It was like, "Oh, that's cool. Let me just use that rather than rewriting it all myself."

So what this does is it takes each of the entrants of the graphics contest and puts their name on a card. And those cards are shuffled and displayed on the screen. So what we're gonna do is pick one of those cards, and then this will gradually go through and reveal all the other cards who are the non-winners unfortunately. So you can sort of – I know it's kind of brutal, but that way you've got a sense if you're still kind of in the running as we go along. So now, I need to pick a card. These are all random, so it doesn't really matter which one I pick, although afterwards it'll be like why didn't you pick the one to the left. Oh, let's just pick this one. This one good?

Student: Yeah. [Inaudible].

Instructor (Mehran Sahami): Yeah. All right. See? Here's everyone else. They get revealed slowly. I don't know if you can hear the sound effect of the cards flipping over. Sorry, Jasmine. You don't win again, but you still got a prize up here. Yeah, some of the cards once they're all covered up – well, Paris doesn't win again.

And the winning entrant is – [inaudible] actually did a very nice painting program. I don't remember all the entrants. And you have your choice of the Three Musketeers or the [inaudible] bar. No one likes the Three Musketeers. I should write that down. I'll remember it. Send me e-mail – also gets 100. Very nice work on the graphics contest. Thanks very much. And now with that bit of business taken care of, we can move on to reviewing for the final exam. So a little bit of final review, what you need to know, what you don't need to know, and maybe a couple examples of the level of difficulty we expect.

So in terms of what's actually on the final, this is kinda what's on. And at a highest level, what's on the final is Chapters 2 through 13 of the book. Okay? So Karel is not on there, and Chapter 1 on history of Java not on there either. Now 2 through 13 covers a bunch of stuff, so in relation to that, let me start off by telling you what's not on the final so you get a sense of what are some of the things you don't have to spend your time on. Then I'll spend some time emphasizing some of the things you do wanna worry about studying. So history, the Chapter 1 stuff as I just mentioned, not on the final. Karel the robot not on the final. He's fun, but we're sorta done with Karel at this point. Zelda might be on the final. No, just kidding. Something people have been wondering about, stack heap diagrams not on the final. So stack heap, the diagrams are not on the final, but you might still be expected to know what is the difference between a stack and a heap, what gets allocated on the heap versus what gets allocated on the stack, but in terms of actually drawing the diagram with the memory addresses, that's not something that we'll ask you a question on.

In terms of the chapter that has to do with images and graphics, there's a whole bunch of stuff if you read the whole chapter about bit operations on images. Bit operations not on the final, so if you sort of skipped that question, we didn't talk about it much in class except where we did a simple example where we took a picture that was color and turned it into black and white. Remember that? On Halloween, no less. I remember the lecture. Bit operations not on the final exam. You don't need to worry about that. Something we sort of stressed the whole time in the class which you didn't need to worry about was polar coordinates, so polar coordinates it's not like hey, we're gonna whip them out in the final exam, now you need to know them. You didn't to worry about them before. You don't need to worry about polar coordinates with respect to the graphics, right? And that's the only place they actually get used. Somewhere else in your life if you go exploring the poles or something, you might care about polar coordinates, but not in this class.

So other things you don't need to worry about: layouts. We talked a bit about layouts when we talked about things like the border layout, and the table layout, and all this kind of stuff. And I showed you could lay things out where you have a program that has a text area for the console, and a canvas, and all that. You just don't need to worry about layouts at all. That's not something we'll ask about.

Sorting – so Ben gave a nice lecture on sorting. Sorting's kinda useful. These days, sorting is mostly a solved problem. We're not gonna ask you any sorting questions on the final in the sense of you don't need to know any of the special sorting algorithms. We may ask you to keep something sorted, but if you think about keeping something sorted, that's different from sorting it from scratch. So we may or may not ask you to do this, but in terms of being able to sort something from scratch, like knowing selection sort or insertion sort, those algorithms you don't actually need to worry about them. The stuff we did on advanced topics where we talked about threads, you don't need to know. It was an advanced topic. It was for your own edification. Some of you used it in graphics projects, which was – or graphics contest entries, which was nice to see, but you don't need to know it. We're not gonna do any kind of testing on threads. Okay? And after threads, there was also a lecture on standard Java where we talked about the main method, and generating jar files, and all this nuts and bolts sort of stuff. That's not something we're gonna test you on either in the sense of standard – we'll still test you on Java. That doesn't mean you don't need to know any of Java, but just that specific stuff we talked about in relation to standard Java versus the ACM libraries, that's stuff you don't need to know about either. So what do you need to emphasize when you actually do your studying? Here are the topics, so if you pop all these topics out of Chapters 2 through 13, the rest of the chapters are pretty much fair game, and here are some of the topics that generally come up which you wanna make sure you're good with.

So the most basic thing is you need to be able to write programs, so you need to be able to build a program. That means knowing all the basic syntax that we talked about, what a class is, extending console program, if while loops, for loops, all that kind of stuff, you're just gonna need to know that. It's not like the exam will not have while loops in it, like do the exam without while loops. It's not gonna be like that. You should know also about objects, classes, and interfaces. What does it mean to implement an interface? What does it mean to define an object? What's a constructor for an object? What goes in a constructor for an object? Stuff like that. So you should be able to – if somewhere on the exam I asked you to define a class that does something, or perhaps define a class that implements a particular interface and show you what that interface is, you should be able to do that. Note on your sample – on your practice final, there actually is exactly a question of this form where you write a class that implements an interface so you get a sense for what that actually is.

You should know about the mechanics of method calls. And part of the mechanics of method calls is what kind of things get passed by reference, meaning they get changed once you actually pass them, what kinds of things get passed by value, which means you get a copy and they don't change, and how those interact with what's actually going on when you make a method call. And so related to that is the notion of primitives versus objects. What are the primitive types? What does it mean for something to be an object? The general generic type object that all things that are objects are a subclass of the type object, so if you wanna write something that is entirely generic, you could write it in terms of being able to keep track of an object for example. And you did this to some extent when you did for example breakout. There instead of an object you had a GObject that you kept track of. What was the thing on the screen for example that you collided with? You wanted to see if that was a brick, or the paddle, or whatever? Hopefully, you did that for breakout but that's kind of the same idea there.

So objects, classes, all that happy stuff – this is now in the what is in the final rather than not on the final, so I'll erase all the not on the final stuff. Other things that are also on the final, strings and characters. Know your strings and characters. Especially in relation to strings and characters, know your string operations. This is just a good thing either to have some tab say in your book to where the list of string operations is, or to have it on a separate piece of paper, or to print out for example lecture notes that are on the class website that contains a list of the standard string operations. It's just good to have them as a reference because you will invariably be doing at least one problem perhaps more that involves some kind of string operation manipulation, and unless you have them all memorized, if you have some handy reference, that's probably a good thing to have. And related to string operations is also – and dealing with characters, so the fact that you can pull individual characters out of a string, you can concatenate a series of characters together to build up a string which is a common way of actually building up a resulting string. That's also kind of related to that, so you need to know the interplay between characters and strings. Graphics, yeah, there's probably gonna be at least one question on there that involves graphics, and that's in terms of the ACM library, so all your happy friends like GRect, and GOval, and GCompound, and all that kind of stuff could potentially show up on the final exam. Again, this is probably somewhere in your book where you have all the different methods listed out for the ACM graphics library, it might be worth putting a little post-it or something in there so you can flip to it quickly if you need to.

And related to graphics is event-driven programs, and by event-driven programs – we've looked at two kinds of event-driven programs. One is event-driven programs having to do with mouse inputs, so what we refer to as a mouse listener, like was the mouse moved, was the mouse clicked, all that happy stuff that you did for example in breakout. And then the stuff that you've been doing more recently with name surfer and also with face pamphlet which are action listeners which are various kinds of buttons, or text fields, or whatever the case may be. So you might have some program that actually for example uses both of these things that may take some mouse input or whatever and does some modification on graphics as a result. Uh huh?

Student: [Inaudible].

Instructor (Mehran Sahami): Ah, excellent. I was just about to say that, and you raised your hand first. No, you don't need to worry about key listeners. So anything we give you will be in terms of mouse listeners and action listeners. You don't need to worry about key listeners.

So other kinds of stuff – then we got into the world of data structures, and we had arrays and array lists. These are also fair game for the final, so some things you should know is the differences for example between arrays and array lists. When is appropriate to use one of them versus the other, so the use of them, or use them. They're your friends. In the sense that if I give you a problem that involves – I give you kinda the setup for that problem, you should be able to choose is that appropriate for an array or an array list. I might tell you which one you should use, but if I don't, you should be able to determine what's appropriate.

In terms of arrays, you also do need to worry about multidimensional arrays. You did some of this presumably already on the Yahtzee program, but multidimensional arrays – at least within reason. I won't give you anything like a 16-dimensional array, but probably two dimensions is fair game. Three is the upper limit. I wouldn't give you anything more than three dimensions. Likely if there's something with multidimensional arrays, it'll probably be two at most. Maybe it'll just be single dimension arrays. And also with array lists, everything that we've done in this class, which is also everything you need to worry about for the final, is sort of the Java 5.0 or later version of array lists. So in the book there's always sort of two versions. There's the here's how you use array lists before Java 5.0 and here's how you use array lists after Java 5.0, or 5.0 and later. You don't need to worry about the before 5.0 stuff. Just the 5.0 and later is all you're gonna need because that's all we've emphasized in the class, and that's these days what everyone's using anyway.

And then after arrays and array lists, which was kind of the introduction to the whole notion of Java collections, was thinking about collections. So besides just array lists you also saw hashmap, which hopefully at this point you are just the master of the hashmap, but you should know the hashmap. And things that come along with hashmaps and collections in general, or the collection interface – I should say collection instead of collections – is for example an iterator. What is an iterator? How do you get an iterator from a collection? How do you use an iterator? That's something that you just wanna be facile with because chances are for some problem to solve the problem, especially if you're doing something that involved a hashmap, you probably wanna be able to get an iterator. And then hashmap you've gotta know, "Oh, hashmap, I need to get the set of keys." Right? The keyset to be able to get the iterator on that. So these are also good places to have tabs in your book for quick reference if you need them.

And last but not least, files. So the basics of files. And you've done a lot of the basics of files already. Know how to open a file and read from a file. Know how to be able to write to a file if you need to write to a file. Chances are we probably won't have you write to a file since you didn't have to do it on any of the assignments, but at least reading from a file is fair game. So topically, that's kinda what's going on there. Now given the topics, let me show you some examples of the kind of questions we might ask besides what's on the practice final which you already have sort of a whole practice final that you can do, and I would encourage you to do that under a timed condition, so you can figure out what kind of things you're maybe a little rusty on, and what kinds of things you're feeling good about.

Here's an example of the kind of question we might give that involves a bunch of these concepts, which is given a hashmap – so you're given some hashmap whose type is hashmap string to string, and you know that these are keys and values. Find essentially all key values are the same. So what we want you to do is find matching key value pairs. What do I mean by that? What I mean is let's say we have some hashmap where here are the keys over here, and here are the values over here. And I might have some keys like Alice, and Bob, and Cathy. And I might have some values like Alice, Jeff, and Cathy. And so what I wanna find out for example to print on the screen is I'd wanna print Alice, and I'd wanna print Cathy. I wanna write out any of the keys whose value is basically as their value in the hashmap. So that's a very simple sort of set up, but it stresses a bunch of these concepts that you need to know to be able to do it. And that's kind of hopefully the way a lot of the questions are gonna be on the final is simple concepts but stresses the ideas. So if we wanna think about solving that problem, we might be given some header like public void match keys, and this has some parameter that it's passed like a hashmap from string to string that we'll call map. And all this syntax should hopefully be stuff that's just second nature to you at this point. You really wanna figure out how to solve the problem.

If you think about this problem, if we wanna find out which keys have the same value as their value, what's the first thing we wanna do? Run iterate over the keys, right? We wanna have some way of saying, "Okay, I need to go through all the keys and look at them to see if they have the same value as their value." So I'd need to have some iterator. The iterator I know would have to be over strings because it's gonna be an iterator over the type of whatever the key is for the hashmap, and I'll call it it. And how do I get an iterator from the hashmap or over the keyset of the hashmap? So I need to say map dot keyset – these are good things to know in the back of your head – iterator – we sort of string these all together, and so what I now get is an iterator of strings which are all my keys in the hashmap. And now at this point it's just using my iterator but in conjunction with some other functions, like while my iterator has a next – and if you wanna use sort of the funky for each notation that's also in the book where it allows you to do iteration over an iterator in kind of a stylized form, that's fine, too, but we didn't stress that in class, so I'm not gonna stress it here.

While the iterator has a next element, I just get that next element – equals it dot next. And then I wanna see is this key equal to the value. So now it's gonna stress another hashmap concept which is to go look something up in the hashmap, and so I have string value equals map dot get, and what I'm gonna get from the map is whatever matches that key.

So if my key happened to be Alice over here, I'd go and get the value which is Alice out of the hashmap, and I'll just keep writing the rest of it over here. We're almost done. So over here we just say if the particular value that I got happens to match the key. Now it's gonna stress the concept of strings. So all these kinds of things fit together. If value dot equals key, which means that value equals the key value, then I print out to the screen to the key. I could print out the value, too, because they're both the same at this point, but that's the whole deal. That's the end of my while loop. That's the end of my method. So that's kinda the idea, right? This was not particularly earth shattering, but it stresses a bunch of concepts. Unless you remember the little things like, "Oh yeah, I need to get the iterator over the keyset," which is kind of a very standard pattern with hashmaps, it could take you a lot longer than it otherwise needs to take you. Okay, so any questions about that?

Let me show you one other simple example that's the sort of thing that would be fair game which is a little program called target seeker. Basically the idea here is there's a little target that's a red square, and there's a little target seeker that comes after it which is the black square. And the black square's trying to get its center to match up directly with the center of the red square, fairly straightforward. And every time I click the mouse, I move the target somewhere else, so I'm just keeping the seeker – it's like, "Oh look, I'm so confused." Every time I click the mouse, the center of the target moves to the current mouse location, and the seeker just keeps seeking to wherever that target is. This problem won't be on the final, 1.) because I'm showing it to you now, and 2.) because it's really difficult to actually put that in a handout. Like take these set of pages and just flip through them really quickly and you can see what the seeker is doing. So how do you turn this into code? And the code for this is actually fairly straightforward. It's kind of you need to think about what you were doing in breakout, right? So some of the problems that you're gonna solve leverage concepts that you've seen in your programs, so you've had experience with these kinds of things before.

What we do is we initialize the target. Let me just line up these lines so it looks a little nicer. The target is basically just a rectangle whose size is some constant target size that's defined down here. I'll just show it to you. Target size happens to be ten, seeker size is twenty, and there's a pause time of ten, just so you know. So target square is gonna be the target. Target square is gonna be some new GRect that we set up, and that's a local – or a private instance variable that I wanna keep track of. My target square is just a private GRect. My seeker is also gonna be a private GRect, and I'm gonna keep track of the target that I wanna get to in a private X and Y location known as target X and Y.

So when I start off, I just set my target to be a square of this size, its color is red, and it's filled to be true, so it just initializes the target. And now I say, "Hey, the target starts in the middle of the screen." Presumably that would be in the instructions for the assignment, and so I get the middle of the screen. Know your standard ways of computing the centers of something. That's always good to know. Then I add my target to essentially the center of the screen, which means if I remember that my target square, the place I add it to a canvas is actually its upper left hand corner, I need to take the target X which is the middle of the screen, and subtract off half the width of the target in both the X direction and the Y direction to get its upper left hand corner. And then I add it to the screen, so this just sets up the target.

Then I set up the seeker, and the seeker is also a square of size seeker size by seeker size, and it just starts at zero, zero. That's presumably something that would be given to you in the problem, that the seeker just starts at a different location. And once I've set up the target and the seeker, I add my mouse listeners because I need to know where my mouse clicks, and then I just have this loop that's just always seeking. It's just always trying to find wherever the target is. And the target starts off at this target X Y location. That's where it always – how we're always gonna keep track of it. So what happens is on every step of seek we pause just so the seeker doesn't just jump there. It's not very exciting if the seeker just jumps there every time. And we say is the midpoint of the seeker – let's get that. That's just the seeker's current X location, which is upper left plus half the size of the seeker to come in. And we would do the same thing on the Y, which means we would go down half the size of the seeker in the Y direction. If my target X is greater than my current X, then I need to move positively in the X direction. Otherwise, I wanna move negatively in the X direction.

And similarly with the Y coordinate, right? If my Y – so I find the seeker's midpoint Y. If my Y is less than the target Y, I need to move toward the target Y, so I'm gonna increase Y. If my target Y's less than the seeker's Y, I'm gonna decrease Y. So I just find these offsets. Are you gonna move in the positive or negative X direction, or are you gonna move in the positive or negative Y direction? And then I'm gonna have the seeker move one step in that direction. Uh huh?

Student: Why'd you do that instead of just finding the location [inaudible]?

Instructor (Mehran Sahami): Having it move consistently with –

Student: [Inaudible].

Instructor (Mehran Sahami): Well, so if you actually wanna do that, if you wanna have it look more nicely so that rather than trying to match up the Xs and then the Ys, if you wanna move toward it, you need to compute angles, and I just didn't wanna deal with polar coordinates, but you actually can do it with polar coordinates. We just don't do it because we're not – that's another thing that you know. You don't need polar coordinates, which is why you don't need to know that.

And the only time the target gets updated is when the mouse gets clicked. We set the target X and the target Y to be the new location of where we clicked. We removed the old target from wherever its location was, and we add it back to the new location. And that's the whole program. That's kinda the level of difficulty. Now this is a program that when you have the code, I can explain it all in like five minutes or so, but when you're writing this from scratch, I would probably expect that writing this program would take probably on the order of about 20 to 30 minutes to write it from scratch with some false starts, et cetera, et cetera. But this is kind of the level of difficulty you could expect to see on the final exam, and a notion of about how much time we would actually give you for it. So any questions about this or about our little hashmap example? That kind of gives you a sense of a set of topics or the richness in some of the questions. And you sort of got the explicit list of topics as well for the class. So any questions about the final? You're feeling okay. So I wanted to give you a bit of wrap up if there's no more questions, given that this is our last class. So if you kind of think about where we've been, we sort of had ten weeks – a little less than ten weeks if you count some – oh, we had a day off after the midterm, and we had a day off on Friday, and whatever the case may be.

But I would imagine for a lot of people in this class, there wasn't necessarily – there was some previous programming experience because I went through all your Assignment No. 1 e-mails. Remember those? That was so long ago. There was some prior programming experience, but there wasn't a lot of programming experience. So the real hope in this class was to get you to a point where basically you could take this journey over here from not necessarily doing any programming to getting to a point over here where now a bunch of the stuff that you actually see, like games that you actually play either online or in an arcade if any arcades even exist anymore, like on your Xbox or whatever, say like Zelda.

For other kinds of things that you interact with like a data management system or social network, when it comes down to it, hopefully this class has helped demystify some of those things, that really underneath the hood, yeah, there's a bunch of listeners that are looking for things, and there are some graphics that get updated, and there's little pause loops, and we need to look for mouse events, and keyboard events. There's a bunch of big data structures like arrays, and hashmaps, and stuff like that. They keep track of stuff like your list of friends in the social network, or the communities you belong in. When you use a search engine, this is just doing the same thing. There's just some really huge repository of data that you now need to go look up. And the interesting problems that come up are things like this huge repository of data doesn't fit on one computer anymore. So what do you do? You break it up onto a bunch of smaller computers, and each of them gets some chunk of the data. And now you go from just simple data management problem to something worrying about breaking up data into bigger pieces.

And this is how computer science progresses in the sense that the problems get bigger in terms of what we actually deal with, and the interactions with the user get more complicated. But hopefully along the way what you learned in this journey was the science of what's involved in doing this stuff. So it's just a bunch of science as to how you actually figure out how all these pieces of a game interact, or how data is kept track of in some larger system, especially when you have smaller components that need to be kept track of in arrays and hashmaps that build up into larger components. And so those are the scientific parts of it.

But hopefully as you also saw today when we showed some of the graphics contest examples and when you wrote your programs there was obviously a variety of ways of doing the programs. That was the thing that at the beginning of the class isn't always clear is there's a huge art in computer science, which is why the textbook for the class is called the Art and Science of Computer Science – is this class, we taught you a lot of this stuff, and we hopefully taught you along the way some of the skills to be a good artist and to be a good software engineer. And the real key in programming is you'll learn more of this stuff as you go along, as you become a computer scientist because there is computer science after all in computer science, and so you'll learn more of this stuff. But what you'll also develop as you go along the way is really being an artist.

There's a famous quote by someone named Don Knuth who's considered the father of modern computer science, which interestingly enough he's still alive and he's in this department, so it's kind of like you're in the same department – you should go look him up, actually – with the father of modern computer science. And one of the things he said is that programmers tend to better if they subconsciously think of themselves as artists. When you're putting something together, don't think of it like you're doing a proof in math. It's important to figure out all the logical steps, but at the end of the day, you're putting something together that has an inherent beauty to it. And that's part of the real beauty of computer science. So I hope you enjoyed the journey. I hope you learned some stuff along the way. And I just wanna thank you for taking part in the little journey we had together. So good luck on the final exam.

[End of Audio]

Duration: 47 minutes