Programming Methodology-Lecture01

Instructor (Mehran Sahami):Alrighty. If you could have a seat, please, we need to get started. There are still a bunch of people coming in the back. Come on down and try to find a seat somewhere. If you can't find a seat, sit in the aisle as long as you're not a fire marshal. Anyone here a fire marshal? Good. We're fine. Come on in and sit in the aisles.

So welcome to CS106A. If you don't think you should be in CS106A, you think you should be somewhere different, now is probably a good time to go, not that I would discourage anyone from taking this class. I think we'll have a lovely time in here. But this class is CS106A or E70A, so if you're, like, "Wait. I thought I was in E70A," you're fine. They're the same class; it's the same thing. No worries, okay?

There's four handouts. They're in the back. If you haven't already gotten the handouts because you came in and you sat down, don't worry. You can pick them up on the way out. They're the same handouts. They'll still be there.

So just a quick introduction. That's what the first four handouts actually give you. They give you a little bit of an introduction to the class, what we're gonna cover, some logistics for the class and some other stuff. I'm gonna go over all that today so we can sort of get a good idea for where we're at, okay?

So just a quick show of hands before we get into a bunch of things in the class. This is kind of an intro-programming course; well, it is. I shouldn't say it's kind of an intro-programming course. It is an intro-programming course. And it's always good to get an idea as to how much familiarity you may have beforehand, okay? So just quick show of hands. How many people can recognize a computer that's on? Good, good. That's the prerequisite for this class.

So if you're worried about how much previous experience you've had or your friend who, like, worked their way through high school by programming for Google or whatever, don't worry about it because all you need to know in here is basically either how to turn a computer on or to recognize a computer that's on if you were to walk up to it and it were already to be on, all right?

So but a little bit more seriously, how many people have actually used a computer for anything? All right. I would expect most of you.

So now, we begin to bump it up a notch. How many people have used it for word processing? Okay. Most folks.

How many people have done web browsing? Yeah, I won't ask you what you look at, all right? It's just I don't wanna know.

How many people have actually created a web page? Okay. Fair number.

How many people have done any kind of programming before? Fair number. All right.

How about how many folks have done actually programmed in Java before? All right. A few folks.

How about another language, C, C++, BASIC, anyone program in BASIC? Yeah, oh, I love — that was the first language I learned, and it was kind of like the warm and fuzzy, and I felt good. There was actually people who argued that if you learn BASIC as your first language, you're brain damaged, then you're just beyond help. But if that's the case, we're all in the boat together because I'm probably brain damaged as well. The truth is I probably am, but that's a whole different story.

All right. So one thing you should know kind of up front is actually this course is gonna be provided eventually somewhere down the line as part of Stanford School of Engineering Free Course Initiative, which means not only are we recording this course to broadcast to a bunch of companies and industry who are watching this course, but we're eventually gonna provide it free to the world.

So how does that impact your life? And on the average day, it doesn't at all. The only way it does impact your life is just so you should know, the lawyers told me to tell you that your voice, should you ask a question, may actually be recorded as part of the video. As a result, your voice may end up going out to thousands of people or millions of people in the world. If you have an issue with that, come talk to me. If you don't, everything is just fine, all right?

Don't worry. We're not gonna put your picture up or anything like that. You might wanna be on the video, like, "Hey, ma, I'm on TV." We decided that we're just gonna not show anyone actually on the video, but your voice may actually get recorded, okay?

Now, along those lines, you may also notice there are some microphones in the room. So when you wanna ask a question, please make sure to use the microphone because that's not only good for people in here to be able to hear your question, it's also good for all the folks that this is getting broadcast to because not only are we gonna broadcast to the world, but there's actually some folks who are sort of watching this live now in various companies in Silicon Valley.

So it's real important that you actually use the microphone, so just remember that. And every once in a while, I might get on your case and be, like, "Please use the microphone." I'm not trying to be argumentative or anything. I just wanna make sure we pick up all the audio, all right?

So with that said, a little bit of an introduction. That's kind of a way of background. I didn't give you any sort of introduction. So just to introduce myself, my name's Mehran Sahami. I'm the professor for the class. Don't call my Professor Sahami, way too formal. Don't call me Mr. Sahami. That, I think of my dad. And don't call me Mrs. Sahami, or we're gonna have issues, all right? So just call me Mehran. We'll get along. It's just fine, all right? It's to keep things a little bit more informal, but that way it's a little bit easier to discuss stuff as you go along.

There is also a head TA for the class, Ben Newman, who's standing up there. Get to know Ben. He has all the real power in this class. I'm just kind of the monkey that gets up here and gives the lectures. But Ben really is the one who's got all the power.

Along with the head TA for the class, we have a large section leading staff. So the section leaders here, could you stand up if you're here? They're kind of all over the place, some over here, some over there, and some over there. As you can see, there's a pretty large number of folks. And this isn't even all of them. We sort of have more — we just can't stuff them all into the room — who are section leaders for the class, and these folks are all here to make sure that everyone in this class has as good an experience as possible when we're sort of going through the class.

And the best way to reach all of us is email. So on Handout No. 1, you get my email and Ben's email. We'll tell you how to sign up for section. That's how you'll meet your section leader and get your section leader's email. That will all be coming soon. But email really is kind of a happy form of communication to get a hold of us, okay?

So with that said, I wanna tell you a little bit about this class and kind of what we're gonna do in here and what you should expect and make sure that you don't feel scared off by this class, okay? Because it really is meant to sort of be an interesting time.

But one question that comes up is why is this class called Programming Methodology, right? Why don't we just call this class, like, Programming with Java? And the real reason for that is that programming methodology is about good software engineering principles. It's about something that's much larger than just programming.

So some people, like, they'll go and get a book somewhere and they'll think they learned how to program by just reading the book. And they're, like, "Oh, I know how to program. Isn't that great?" And it's, like, yeah, you might know the mechanics of the language, but the mechanics of the language are nothing compared to understanding the software engineering principles that go into actually developing a software system.

And that's what you're gonna learn about in this class. You're gonna learn a lot of those principles. But in order to be able to use those principles and apply them, you also need to have the language to program in, and that language that we're gonna use in this class is Java.

So the way I like to think about it and the way I tell a lot of people is writing a good program or learning how to program is like learning to be a good essay writer. And you're, like, "Oh, but part of the reason I'm taking this class, Mehran, is that I don't like writing essays." That's fine. It's okay. Trust me. I didn't like writing essays either.

But the whole point is that when you write an essay, it's not a formulated kind of thing. You're, like, "Well, what about five-paragraph essays?" Yeah, just block that from your mind. That was a bad time, right? That was just, like, '70s education at work. It's not a formulated kind of thing.

There's an art to writing an essay, right? In order to write an essay, you need to know a language. You need to know English or German or Hindi or whatever language you wanna use, but then you use that language to write an essay. Just knowing the language doesn't make you a good essay writer though. Being a good essay writer makes you a good essay writer.

So that's the same difference in programming and software engineering. Knowing the language, in order to be a good programmer, like a good essayist, you need to know a language to write your programs in, whether that be Java or C or C++ or whatever. Here we're gonna use Java.

But just knowing the language doesn't make you a good software engineer and doesn't make you understand what the principles are of writing good software, which is what you're also gonna get in this class in addition to the language, and that's kind of a key thing to stress.

So if you're sort of worried, if you were kind of looking around and you saw a bunch of people raising their hands when I asked, "Do you have any previous programming experience?" and some folks raised their hands, and you got a little worried and you're like, "Oh, am I gonna be in some sense at a disadvantage because I haven't done any programming before?" The answer, plain and simple, is no, okay? You're gonna learn everything you need to learn from the first principle because as a matter of fact, in some cases you might be in slightly better shape. That's not necessarily to say that that's the way it will be.

But how many people are Star Wars fans? Just wondering. Anyone? I'm talking about the old-school, original, like, three movies. Those were so good, and we're not — no George R. Binks here, all right? So if you remember — and sort of I'm a big Star Wars fan, and that's just a whole separate point. But in the second movie, Yoda actually said something which I thought was quite profound, which is he says sometimes you have to unlearn what you have learned.

And one of the things we actually find is that some people who are self-taught programmers, some of them are just fine, and some of them are very good. But some of them have picked up some really bad habits along the way, and it's like being a bad essay writer. And to go from being a bad essay writer to a good essay writer, in some cases, can actually be harder than from not being an essay writer to being a good essay writer because you have to unlearn the bad habits.

So if you're worried about, "Oh, I've had no previous experience," don't worry. You're okay, blank slate, you're just fine. And now if you're thinking, "Oh, I have some previous experience. Do I have bad habits?" Don't worry. You'll be fine, too, okay? So it's all gonna work out.

So the next question that kind of comes up — hopefully that helps put some of your fears aside. Another one of the things is that we really strive to make everyone successful in this class, okay? At some other schools, people wanna do computer science or they wanna do an engineering major or whatever. And you come into the first day of class, and they say, "Oh, only one third of you are actually gonna make it through this program. And look to the person to your left and look to the person to your right, and only one of you will make it through." And you're, like, "Oh, man, that's real nice." It's not like that here.

As a matter of fact, we want all of you to be extremely successful in this class, which is why we have a huge course staff, which is why over years and years we've refined how we do a lot of the teaching in this class to make sure you have the best possible experience and to make sure that everyone gets through.

And the important thing about that is that you're not competing against anyone except yourself in this class. It's not like we're gonna have a curve and we're gonna say, "Oh, we have a certain number of "F"s and a certain number of "D"s and a certain number of "C"s." All we really have going into it is an expectation that when you get out of here, there's a set of stuff we want you to know. And if you know that stuff well, you get an "A." And if everyone knows that stuff well, everyone gets an "A." And I got no problems with that. Registrar might have a problem with that, but that's okay. You don't need to worry about that.

So you don't need to think about, oh, is someone else doing better than you or whatever. And we'll talk about issues of collaboration in just a little bit. All you need to think about is learning the stuff yourself as well as you possibly can, and you'll be just fine and you'll get a good grade, okay? So that's really all we ask, which is not a trivial amount, right? It requires you to really understand the material.

So another question that comes up is are you in the right place, right? This isn't the only introductory programming class at Stanford. And so I wanna spend a little bit of time making sure you actually are in the right place by going over some of the different options.

So right now, as you know, you're in CS106A. And CS106A, we're sort of happy over here, right? As a matter of fact, we're not only happy, we're happy and we're also a little bit loopy, right? There is no previous programming experience required, as I mentioned, right? All you need to know is basically if you can get to a computer and know how to figure out that it's on, you're in good shape.

But what 106A does is it's a real rigorous class. You learn programming in here, and you learn it in a way that makes you ready to be an engineer if you so choose to be an engineer. That's not to say you're all gonna be engineers. I would love for all of you to be computer science majors, but statistics in the past show only about 6 percent of you will be computer science majors. That's not because we turn anyone off to computer science; it's because we make programming accessible to so many people that you don't have to be a computer science or a Double E or even an engineering major to do extremely well in the class.

And we actually have sort of a significant percentage of the entire campus undergraduate student body at Stanford actually goes through this class and does well, okay? So don't worry if you're, like, "Oh, but I'm not really a CS person." I hope we'll turn you into one by the end of the class. No, it's okay. But you'll be prepared if that's what you wanna do. So this leads into a whole engineering sequence that can go on to other engineering majors or the computer science majors.

If you're, like, "Huh, I'm not sure if that's really what I wanna do. As a matter of fact, I'm so sure that's not what I wanna do, I only wanna get the general educational requirement out of the way, and I'm positive there is nothing else I wanna do. Really, no matter how much I like it, like, there is no way you're gonna drag me into anything that would involve anything remotely techie." They're the class CS105. And this is happy, yeah, this is kind of, oh, we're happy in our little happy world. And I don't wanna say it's holding hands and singing, "Kumbaya," because that's not what it is. It's a real class. But it's meant to be a general educational requirement, right? It doesn't lead into the 106s. It's meant to be its own self-contained class. You do some Java script in there. You do a little bit of what computers are about.

Computers in society is a good time. We all hold hands. We're all happy. I don't teach the class, so I don't actually hold hands. But it's a fun time, okay? It just doesn't lead to anything else. So think of this as kind of a terminal class, right? So it's sort of like, well, we'll hook you up to the IV drip.

And you're, like, "Well, 106A, you told me I don't need any previous background. Well, hey, Mehran, I got lots of background. I got so much background, it hurts. I got AP background, I got working through school doing software engineering background. I'm not sure I should be here." That could be the case.

We have another class called CS106X, and as the "X" kind of implies, it's sort of the extreme games version of the class. No, it stands for accelerated, right, because "A" was already taken, so we had to come up with something else. So the way CS106X works is it really is a very fast-paced class. It's meant for people who've got previous AP exam credit, like, got a 4 or 5 on the AP, or have had significant and prior programming experience before.

If you're not sure which one of these classes is for you, you can come talk to me afterwards, or I'd also encourage you, you could go to pick up the syllabus for CS106X and compare it to CS106A. This class is all in C++. And if you're thinking, "Hey, Mehran, I'm doing 106A. I wanna learn Java and C++," don't worry. You'll eventually, if you so choose, take a class called CS106B, which is where this class sort of leads to, which is C++ and all of the other stuff you would have learned in this accelerated class, okay? So you still certainly have that course path.

So don't let anyone make you think — I know a lot of times, and especially for Stanford students, you come in here and you're, like, "Well, every class I took in high school was like an honors or an AP class, or if it wasn't an honors or an AP class, like, I had to tie half my brain before my head because I'm just that hardcore." And so everyone just wants to, like, do the most hardcore thing they can, right? And what I'm here to tell you is that you shouldn't necessarily think about it that way. You should think about it as where you feel most comfortable.

Some number of years ago, let's just say greater than 10, maybe 15, I was sitting where you're sitting right now, literally. I was in CS106A in Terman Auditorium as a freshman, okay? It was perfectly fine. It worked out. I went to grad school, did the faculty thing. It's just fine. It will open your doors to CS. You're not at any kind of disadvantage by starting here. So know where you've been, literally. Like, that seat right there was where I was most of the time. So just something to keep in mind in terms of the different options that are actually available to you.

Now, with that said, let's just assume for the rest of this lecture that this is the right place for you. And if it's not, well, afterwards we can kind of talk about it, or if you really are convinced now that it's not the right place, you can feel free and try to scramble over 20 of your classmates and actually leave the room, which is probably impossible.

All right. So a few other things you should know, some mechanics. So Handout No. 1, should you wanna follow along at home, is the class web page. And so all the stuff that we think of as course materials, including online copies of the handouts, things that you'll need to do for the assignments, announcements related to the class are all on the class web page, which is www.stanford.edu/class/cs106a. And because that's just kind of a whole bunch to remember, we make your life easy and so there is an equivalent form of the URL, which is just cs106a.stanford.edu, which is the easy thing to remember. You put that in, it'll take you to the class web page, okay?

And you should check that regularly because all the announcements and handouts — we'll give out hard copies of all the handouts in class, but should you happen to miss class for whatever reason, you wanna go print whatever copies of the handouts we're actually giving out, you can find them all on the web page, okay?

Now, there's this funky thing about units. So you may have noticed that this class is for three to five units, and that kind of brings up the natural question, "Should I take it for three or five units?" If you're an undergrad, you take it for five units, end of story. That's life in the city. Congratulations. Five units.

If you're a graduate student, you can have the option of taking it for three units if you want, if you're gonna run into some unit cap. It doesn't change the amount of work you have to do. Welcome to graduate school. Same work, fewer units. So that's just the way life is. If you have a unit cap and you're a grad student, in three units you can take it if you want. You can take it for five if you want as well. If you're an undergrad, you take it for five, all right?

So why is it five units? And you might think, "Hey, this class only meets three times a week. How come it's five units?" Well, it actually has a fourth meeting every week, which is your section, and that's something you should sign up for. So how you actually sign up for your section is sections are at a bunch of different times. You don't sign up for them in Axess, even though they're all kind of listed in the time schedule. That's not where you sign up for them. In Axess, you just sign up for the class.

How you sign up for a section is you go to a website, cs198.stanford.edu/section and this will give us a list of preferences for section times that you wanna sign up for, and there's some matching process that goes on. It takes all your preferences into consideration with the whole system, and eventually you get an email by sometime early next week that tells you what section you're in. And section's 50 minutes, once a week. It's required to go to. It's actually gonna be part of your class participation grade, which we'll talk about in just a bit, okay?

When do these sign-ups happen? They happen between 5:00 p.m. this Thursday is when they go up. So if you try to go there now, you can't sign up. Remember 5:00 p.m. Thursday. So they're up, and then they're down at 5:00 p.m. on Sunday, okay? So make sure you sign up probably this weekend. If you're planning on being out of town this weekend, you wanna sign up before you go. Sign up early, but don't sign up often because you only need one section, okay?

If you're an SCPD student — every once in a while you'll hear me refer to SCPD students. That stands for Stanford Center for Professional Development. They are the folks in industry who actually take this class via broadcast. If you're an SCPD student, you're automatically enrolled for a section, so you don't actually need to do this, and your section will meet at — so for SCPD — and if you're wondering what an SCPD student is, you're not one, okay? So SCPD section meets Friday from 1:15 to 2:05. It meets live, if you wanna go there live, in Skilling Auditorium. But if you're watching it remotely, it meets on Channel E2. I know it seems weird to say it meets on channel — what does that mean? It meets on Channel E2, okay? That is grammatically the correct way of saying it.

All right. So there's a little bit more administrative stuff. Now, textbooks, right? Textbooks, there's nothing quite like the extortion that is textbooks. So there are two textbooks that are required for this class. Well, one's a course reader and one's a textbook.

The course reader is called, Karel the Robot Learns Java. You can pick it up at the bookstore. It's relatively cheap. It was actually written by Eric Roberts here. And surprisingly enough, the textbook for the class was also written by Eric Roberts, The Art and Science of Java, which is available now in your local bookstore, including the bookstore on campus, so you can go and pick up a copy of this.

So both these things you actually wanna have because they're required for the class. We'll go through all of them. We'll go through basically everything except the last chapter of this book. So you sort of get your money's worth. We're just gonna do it a little bit out of order, but we'll go through the whole thing, okay?

So email, how many of you have email accounts? All right. I will ask the reverse question because I think at this point, some people just don't wanna put up their hands. How many people don't have email accounts? Odd how that is not the complement of the folks who had their hands up previously. Email's required for this class. Chances are, by being at Stanford, you've already gotten an email account through your SUNet ID, but if you don't have an email account, get an email account and that's how you'll stay in contact with us. That's how we'll stay in contact with you, except we'll also meet with you live in person, but email is kind of the general method for communication.

As a matter of fact, for your first assignment, and part of your first assignment is to send us an email, just because we love you and we don't get enough email as it is. So you need to have an email account to be able to do that. So if you have not already, you can kind of get ahead of the game and go set up your email account. Now, don't worry. You'll get the first assignment next time. So you still get, like, two days of breathing space before your assignment goes out, okay?

There is also gonna be lots of handouts in the class. They'll be either given out in class, well, they will be given out in class, but we'll also post them online in case you miss them.

And how much real work do you do in this class? That's always kind of an interesting question. So let's talk a little bit about assignments and a little bit of other logistical things. So assignments, we'll just call them the dreaded assigns. There are seven programming assignments. And if you look at the syllabus Handout No. 2, it tells you when all of them are due all the way through by day, so you can plan out your whole quarter. It's just that much fun, okay?

And these seven programming assignments are weighted slightly more toward the last assignments because the assignments will tend to get more complicated. That doesn't necessarily mean there'll be more programming; it just means conceptually, they'll become more complicated, so we tend to weigh them more toward the end of the class. So the later assignments count more than the early assignments.

How you're gonna be actually doing your programming is using a little tool called Eclipse. And Eclipse thankfully is free, so you don't have to pay for it. As a matter of fact, you can download it from the CS106A website. And if you're wondering how you do that, don't worry. We'll give you a handout next class that explains to you the whole grueling process of downloading and installing Eclipse.

And you can use this either on the Mac or the PC. So if you have your own computer, you can certainly work on this yourself. You just download it to your own machine. We'll explain the whole process in a handout. If you don't have your own computer, the public computer clusters on campus will have Eclipse installed on them, and so you can use Eclipse there. So you're sort of happy to go either way, okay?

Now, the important thing, remember I mentioned that whole notion of software engineering in the class, and that's something we take really seriously, so seriously as a matter of fact that when you turn in your assignments, one thing we could do is we could take your assignments and we could just kind of look at it and go, "Yeah, interesting, 'B.' Here you go. Thanks for playing." And you don't learn a whole lot from them. So in order to actually learn a lot from your assignments, we could take your assignment and write a whole bunch of comments on it and hand it back to you. Even that's kind of not enough.

What really is a little bit more that makes it more fun is every week after you turn in your assignment and your section leader looks it over and grades it, you'll actually meet with your section leader for about 10 to 15 minutes every week or every time an assignment is due to actually go over in something referred to as interactive grading.

And it's a chance to sit there and talk with an actual human being about what's good in your assignment, what are some of the things you need to work on, what are some of the software engineering principles you need to develop. And that way, you can really sort of get more detailed information and be able to ask questions to develop yourself as a programmer as well as get help if you need help, okay?

And that's in addition to going to section, going to class and all that stuff. So it's another 15 minutes a week. You'll actually schedule that time with your section leader on a regular basis when you're gonna have interactive grading or just affectionately referred to as IGs because at Stanford, everything's just short and we just can't say, like, psychology; we have to say psyche. So it's IG. Just remember that, all right?

And then how are these things graded? So the other thing we could do is I told you we could just write "B" and hand it back to you. But we found that that's not really great because people get all wrapped around the axle about the grade.

And so for a while, we did numbers and we're, like, huh, why don't we give a number between 1 and 20? And so what happens there? People get all wrapped around the axle about numbers.

So then we thought, huh, what was a happier time when we were in school? I remember when we were in school, and we used to get back assignments and they had, like, smiley faces on them. Well, we can't do that because then it doesn't appear to be a rigorous Stanford class.

So instead of the smiley face, we come up with something else, which looks surprisingly like this. It's kind of involved to actually draw, so I need to erase the board to do it. Check. That's kind of the beginning of our grading scale, okay?

And the way our grading scale works is we start off with a check in the middle, which says this is a pretty solid program. It meets all the requirements for the program. Maybe it's got a little problem here or there, but it's a check. Then we have sort of two grades on the two sides of it: check plus and check minus.

Check plus is, like, solid. You did a great job; you got everything right; things look good, a nice style in your program, nice software engineering, and the program works flawlessly. Good job. This is like total "A." Check is kind of like, yeah, you're sort of there. It's kind of like "A" minus, "B" plus, maybe on some occasions "B." But it's kind of like it's pretty good work; you're in pretty good shape here. And so a lot of grades in this class ends up being check pluses and checks, and if that's the case, you're perfectly fine grade-wise.

Check minus, as you can imagine, this is kind of thinking about "B," "B" minus. It's, yeah, there are some slightly more significant problems with your program.

But that's not where it ends, right, because we wanna be able to even shoot for in some sense bigger gustoes. There was a plus and a minus. So plus is like, oh, nice job, kind of a hearty pat on the back. If you get pluses all the way through on all your assignments, you're in a pretty good candidate to get an "A" plus.

And minus, like, just take good over here and replace it with bad, it's kind of like, oh, bad times, right, or maybe, you know — but even there was, like, more significant problems with this program or just the style on the program is just really bad. But even there, we don't stop. And you're, like, "Come on, man. Like, I thought the whole reason was to simplify this." Don't worry.

And it gets even better because we have a plus-plus and a minus-minus. And at this point, we've run out of board space, so we can't go any further. But a plus-plus is just outrageous, right? It's the kind of thing — so this is the kind of thing your section leader can't actually give you without coming and talking to Ben and I first because they get a program that just goes — it has to actually exceed the requirements for the assignment. It's by a long shot. Like, you'll get all your assignment requirements, and what we encourage you to do is you can do a grade assignment and get everything right and have good style, and you'll be in this category. And for the later assignments, you may be in this category if it's flawless.

But we'll actually — if you want to go for the plus-plus, go beyond the assignment requirements. And the way we think about the plus-plus, it's a program that makes you weep in a good way. It's just like your section leader sees it, and they're just, like, this is so good, I've gotta show someone else. And they come and show Ben and I, and we're, like, sitting there looking at this on a monitor, and, like, tears are just welling in our eyes, and there was, like, soft violin music playing in the background and we get out the wine and cheese.

So this is just, like, this is the kind of thing that gets you, like, remembered and the, oh, if you want a letter of recommendation, just ask because you got a plus-plus. Like, oh, it's awesome, right?

There are very few of these in a quarter. So just by sort of way of comparison, in a class this size, probably throughout the span of the whole quarter, I'd expect there to be maybe ten plus-pluses, I mean, ten assignment plus-pluses, not ten students who get plus-pluses across the board. So it's really something to strive for, but if you strive for it, like, we're giving you the credit for it. And this gets remembered and you get, like, extra credit and everything.

So we're left with this, right? This assignment also makes you weep, but not in the good way, right? It makes you kind of weep in the sense, like, I look at them and I'm, like, oh, man, like, what did I teach? Like, where did I go wrong, right? I, like, blame myself. I blame you a little bit, but I blame myself. And this is really just, like, the program is just, like, it's a shell. Like, there really wasn't much effort that was put into it. Yeah, you slapped something together or it doesn't really work, that whole deal.

And then if you don't turn anything in, we do kind of reserve the zero to distinguish from the "made really bad effort" versus "didn't make any effort at all." And we just won't talk about these, right? Let's just hope we can avoid those if possible. But that's kind of how the grading scale works now.

Now, at the same time, I trust all of you to be responsible people. And every once in a while, something bad happens to a good person, and there's an assignment that you'd like to be able to turn in, but for whatever reason, you can't turn in on time. And I just wanna treat you like adults. I don't want you to have to worry about coming in and asking for an extension or, like, "Oh, I had this really hard thing in another class, and I couldn't do it at the same time." Up front, everyone gets two free extensions, okay? So in terms of late days — we refer to these as late days, strangely enough — you get two free ones.

What a late day is, is a class day. They're not 24-hour days, but class days. So if something is due on a Wednesday, you turned in on a Friday, that's a late day. That's one. You turn it in on the following Monday, that's two late days. You can split up your two late days among two different assignments. You can use them both on one assignment.

But we encourage you to not use them at all because if you use your late days, you fall behind in the class. The way you should think about these things are these are pre-approved extensions. They're not the kind of thing where you just think, "Oh, yeah, I'm not gonna do the assignment because I wanna go and play Frisbee golf," right? Think of it, well, you wouldn't come ask me for an extension — you might, but you probably wouldn't ask me for an extension if you're, like, "Hey, hey, Mehran, can I turn in the assignment, like, on Wednesday because I'm playing Frisbee golf this afternoon," right? If you would feel embarrassed asking that question, you probably don't wanna use one of your free late days.

But something happens like, oh, it's a tough week, you've got midterms in other classes and you got this assignment due or whatever, that's a good time to use it. So we just trust you. And most people, we actually encourage you not to use them because it just makes you fall behind in the class.

Because we trust you and we give you these two up front, getting extensions beyond your two free class days is virtually impossible because we sort of up front said, hey, it's your responsibility. We're giving you two freebies. We're not gonna give you a third extension. Imagine if you had to come ask us for three extensions. By the third one, we'd be, like, okay, what's going on, which is why we don't necessarily give extensions beyond these two.

The only time we might give an extension beyond the two free ones is for something major, like death in the family or, like, serious medical problems that might require surgery or something like that. Every once in a while, unfortunately, that happens. I hope it doesn't happen in this class. But those are the only kinds of things that we give extensions to beyond the two free late days.

Importantly, don't ask your section leader for extensions. They cannot grant you extensions. Only Ben, who has all the power in this class, can give extensions, which is why you should get to know Ben and then hopefully you won't need to talk to him about extensions, okay?

So other thing to keep in mind is that three days late is the max. Beyond three days late, which is basically one class week, if you think about late days being class days, we will not accept an assignment. And the reason for that is at a certain point, you're so late, you're better off just doing the next assignment, letting the old one go. So to sort of enforce that policy, after three days, we don't accept that assignment late anymore. It's just gonna be a zero if it's not turned in, okay? And that just kind of forces you to keep up.

Couple other minor things, well, I shouldn't say they're minor things. They're actually kind of important. Exams: There's two exams in this class. There's a midterm and a final. Both of them are, well, I shouldn't say both. The midterm is out of class. It's from 7:00 to 8:30 p.m. on Tuesday, October 30. And it's on the syllabus. It's there. It's on the syllabus; it's on Handout No. 1. We repeat it multiple times. The date will eventually be announced when we get close to the midterm.

But if you have a conflict with this time, you need to send me email, okay? You can send me email a little closer to the midterm because I'll announce it again for people who have conflicts. But since it's an out-of-class exam, you need to send me email if you have a conflict. I'll get all the constraints from people who have conflicts and try to schedule an alternate time if there's enough people with conflicts. But 7:00 to 8:30 is when you need to know about the midterm.

And to make up for the fact that we have an out-of-class midterm, I actually give you sort of a belated free day, which is the Friday of the week of the midterm, we don't have class to make up for the fact that we made you come to the midterm outside of class. But the midterm's an hour and a half, and we can't compress time. If we could, we'd have different issues. We can't compress time and fit it into a 50 minute class, which is why it's out of class, but you get a free day for it, all right?

Last but not least, few things about grading. Grading, one of those things as you might be able to tell from this little board over here or something, if I didn't have to do it, I wouldn't do it because honestly, as corny as this sounds, I just believe in the love of learning. Like, I think if you're passionate about something, you just go do it and you learn it. But I'm naïve, and so that's not the way learning always works. So sometimes we actually need grading to make sure that learning takes place.

And so this is how your grade breaks down: Forty-five percent of your grade is on the programming assignments, okay? Fifteen percent is the midterm, which we'll just call the mid because we like to abbreviate everything. Thirty percent is the final. It's a three-hour final exam in the regular final time slot for this class.

If you think or are under the delusion that you should take two classes at the same time, that's a bad idea because their final exams are at the same time, okay? So you should not take two classes as the same time because our final exam is scheduled for — I believe it's December 13, which is a Thursday, 12:15 to 3:15. That's the regular final exam slot for this class. And any other class at the same time will conflict with that slot. Thirty percent of your grade is the final.

And that, if you add it all up, it's not just that I'm bad with math. It's because 10 percent of your grade is actually participation. And this is things like did you go to your interactive grading sessions? Did you regularly attend section? Did you participate in section? Did you participate in class, right?

And so, in order to help you participate in class, there's a little incentive to participate in class, which is sugar in the afternoon. So someone raise their hand. All right. Yeah, sometimes I'm not a good shot. And this will tell you, if you're sitting in the back of the room, I can't throw a Kit Kat back there because they're a little too light. Oh, yeah, sorry. If you sit in the back of the room, the roof prevents me from actually being able to hit you. So if you want the food, come up. But if you ask questions in class, hey, that's a good time. It's just a little way to be able to reward you for actually participating in class or to keep your blood sugar up if you need it, all right?

So that's participation. It's 10 percent of your grade, and as a matter of fact, at the end of the quarter, I ask every one of your section leaders to actually tell me how much you participated in class, and some of them just say, "Oh, this person was wonderful. They came every time. They participated. It's just a great thing." And that helps your grade out a lot, okay?

Now, the final thing, and as you can kind of tell, most of the time, I'm not the most serious person in the world. I just like to have fun with things, and I think it's important for you to have fun with things. There is just one place where I get real serious, and it's one place where Stanford gets real serious. Anyone wanna guess what that is? Plagiarism and the honor code. As a matter of fact, that's what we call a social. So we had someone down here who got it and then a whole bunch of people who I don't know, so we just spray. All right.

So the honor code, in terms of the honor code, the question comes up is what is the honor code all about, and how does that affect working in groups and computer science, etc.? Does that mean we shouldn't talk to each other? No. The answer to all those is no, okay? If you look at Handout No. 4, which is all about the honor code, we encourage you to talk to each other. We encourage you to talk about concepts in the class, talk about different strategies to problems, to think about the ways that you could potentially approach some problem or the way different control constructs when we eventually get to them work in the class. And discussion is perfectly fine, especially among the course staff, but also amongst yourselves. That's a great thing.

So where do we draw the line? And we try to make a bright line for where you've crossed the line for the honor code, which is don't share code, plain and simple, in any respect, okay? Don't give a file to someone else that's got your code in it. Don't get code from someone else. Don't look at someone else's printout. Don't give them a printout.

If you have two people who are sitting looking at the same screen together, that code can't belong to both of you. It belongs to one of you. I don't know which one, but it becomes an honor code violation. So you shouldn't both — two people shouldn't be staring at the monitor together. If it ever gets to the point where you're looking at someone else's code, that's where you're gonna reach an issue, okay? Discuss as much as you want. That's great. Write your own code. That's all we care about.

And you're, like, "Well, what is code, Mehran? What does that word mean?" Code is geek speak for your program, so when you program, the program that you write is what we affectionately refer to as code. And the idea of programming is what we refer to as coding, strangely enough. Computer scientists need to make everything more complicated than it really is so we can get people under the illusion that they should pay us lots of money to do what we do. I mean, you're, like, "Oh, I just write programs." And they're, like, "Oh, yeah, I should pay you half." And you're, like, "No, no, no. I write code." And they're, like, "Oh, yeah." Suddenly, it's much more impressive. So don't share code.

The other thing is if you talk to other people, like if you have a study group to talk about solution approaches or you go, let's say, talk to the TA or your section leader to how you should approach a problem, and they give you a lot of hints as to how to do it, cite collaboration. So cite and collaboration gets you out of trouble. Any collaboration that you cite you cannot be held responsible for under the honor code.

You can actually copy someone else's program and say, "I copied this program from Mary Smith." And I'll look at that and say, "They cited it," and it will warm the cockles of my heart. And Mary Smith will get full credit, and you'll get a zero because you copied your program from Mary Smith, but it's not an honor code violation because you cited the work, okay? So the bottom line is keep yourself safe and cite your collaborations. And I guarantee you most of the time, you'll be just fine.

Now, you might wonder why do I make such a big deal about this. And the reason I make a big deal about this is for a while, thankfully it's not true anymore, but for a while, the computer science department actually had more honor code violations than the rest of the university combined. Take everything else in the university, put them all together, they were like over here. And we're, like, we're computer science, which is not a fun distinction to have, let me tell you.

And you might wonder why is that? Is that because computer science people are just mischievous and dishonest? No. It's because it's easier to catch honor code violations in computer science. We have a whole bunch of tools that allow us — then we take all your programs and we run them through this tool, and it compares them not only to everyone else in here, but, like, to everyone from the last, like, X years where X is the large number of people who've ever gone through the classes, right? And it's an extremely good tool from finding where honor code violations happen, from where they don't. And it doesn't find spurious violations.

To be honest, I've never lost an honor code case. When I find an honor code case, it is blatant. And you take it to judicial affairs, and they look at it, and they're, like, yeah, this is blatant. And I take it to the student, and every student I've ever confronted them with never said, "No, no, no. I didn't cheat." They said, "You caught me," okay? So it's blatant.

It's not like, oh, there's some little line in it, "Oh, am I gonna need to worry about an honor code violation?" Remember those rules, you have nothing to worry about in this class. It's people who go and, like, fish out printouts from the recycle bins and copy other people's code that are the people we catch, right? It's blatant cheating that we catch. But we catch it. We catch it all the time. So I hope, I pray it doesn't happen in this class.

But the reason I make a big deal about it is historically if I look at the evidence, it happens and we catch it. And when we catch it, we're required by the university to prosecute. And I feel bad because usually it's someone who just made a bad call, like, they were up way too late the night before working on something else, and they're not thinking straight. And rather than just taking a late day or turning in their assignment late and getting a slight penalty on it beyond their two free late days, they decide to cheat. And that's just always the wrong call, okay? So you just don't wanna put yourself in that situation. So I get real serious about it for a moment, and hopefully it won't be an issue and we can just kind of go on, okay?

So with that said, that's a whole bunch of logistical stuff. Any questions about the logistics of this class or anything I just talked about? Uh huh?

Student: You had briefly mentioned the late penalty.

Instructor (Mehran Sahami):Oh, the late penalty, good point. So remember our little bucket scale. If you go beyond your two free late days, every day you turn in an assignment late beyond those, it drops down one bucket. So let's say you already used your two free late days on Assignment No. 1. And on Assignment No. 2, you turned in something one day late and you would have gotten a check normally, it becomes a check minus. So that's how it is. It's one bucket per late day beyond your two free ones. Uh huh?

Student: Are the sections first come, first served?

Instructor (Mehran Sahami):Yeah, the sign-ups, well, they take into consideration your preference, but part of your preference is to do the match is first come, first served. So you wanna sign up early. Oh, thanks for your honesty. As a matter of fact, I dig honesty, all right? Any other questions? It's just honesty's cool. Uh huh?

Student: How much time should we plan on studying [inaudible]?

Instructor (Mehran Sahami):Oh, good question. How much time should you plan? And this is something that I say for classes in general at Stanford, which is not always true, which is take the number of units that a class is, multiply it by three. That's how many hours you'll spend per week in that class, total, on average. So what that means is in 106A, a 5 unit class, you multiply by 3, you get 15. Five of those hours are roughly spent between class, section, interactive grading, other stuff. That means on average about ten hours a week will be spent on your assignments in this class. Again, that's an average.

Sometimes when I go to computer science conferences, I sit there and joke around with plans. And we're, like, "Oh, how long did your assignments take?" And I say, "Oh, on average, ten hours." And what I really mean when I say on average 10 hours is they take between 3 and 45, okay? It's a large variance event, right? Ten is the average. Some people take a really long time. Some people get through it really quickly, but that's about the average you can plan for. Uh huh? Another question?

Student:[Inaudible] late days [inaudible] class days?

Instructor (Mehran Sahami):Yeah, all late days are class days, so the free ones — the halfway mark's really my reach. That's about it. All right.

So I do wanna give you your very beginning of an introduction to programming before we sort of break for the day. How are we doing on time? And so in order to kind of see this, there's a few things that we wanna keep in mind.

Actually, let me show you a little picture, okay? Sometimes when we talk about writing programs, we talk about debugging programs, right? How many people ever heard the term debugging or bugs in programs? A bug in a program is an error in a program, so sometimes when you hear us say, "Oh, come see," like, your section leader to help debug or see the helpers in LaIR.

That's another thing. In the Tresidder computer cluster is the LaIR. It's a computer cluster that we have helpers there to help you get through this class. What is it? Sunday through Thursday, every week, from around 2:00 in the afternoon 'til midnight every day, okay, to help you get through the class. So that's a good place if, you know, you can work in your dorm room certainly, but if you also want help, go to the Tresidder computer cluster, and there will be helpers there. There's a little queue you sign up for to get help, and that's a great place, and it's all explained in Handout No. 1, but that's just something to keep in mind.

Where the term debugging comes from, it turns out this is an apocryphal story, but I'll tell you anyway. Back in the days of yore, in 1945 actually, there was a computer called the Mark II at Harvard. And there was a woman named Grace Murray Hopper. Anyone ever heard of Grace Murray Hopper? A few folks. She was actually the first woman who was an admiral in the navy. And she was also one of the very early pioneers of computer programming. She did a lot of computer programming when she was actually a captain, and she was stationed at Harvard as part of some sort of navy thing. I don't know why, but that's what happened.

And they had this huge computer there, and they were noticing the computer was on the fritz, and they couldn't understand what was wrong. And this is one of these big old machines in the days of yore that has vacuum tubes and stuff inside it. So they walked inside the computer, right, because then you could actually open it up and walk inside your computer.

And they saw this, and I don't know if you can see that, but that's a moth. It was a moth that had sort of given its life to be immortalized because it had actually shorted out across two relays in the computer and was causing these sort of errors to happen on the fritz. And so they took the bug out, and once they actually plucked this little charred bug out of there, the computer started working fine again, and she taped it in her log book.

And this log book's actually preserved in the Smithsonian Institution now, which is where all this comes from. Here's all the standard disclaimer information: "Image used under fair use for education purposes. Use of this image is exempt from Creative Commons and other licenses," just so you know. Now the lawyers are happy. But this is where we think of sort of the modern term debugging actually came from.

Now, it turns out the actual story is that the term debugging came from the 1800s, in the late 1800s from mechanical devices. People actually referred to debugging as fixing mechanical devices. But this is kind of the apocryphal story for how it comes up in computer science.

Now, with that said, what is the platform in which you're gonna sort of do your first debugging or your first work on? We talked about Java, but in fact in this class, we're not gonna start with Java. We're gonna start with something even sort of simpler than Java because as I mentioned, sometimes what happens in computer science is people learn all the features of some language. And they think just knowing the language makes them a good software engineer. And they get so worried about all the features of the language that they don't kind of think about the big picture.

And so there was a guy named Rich Pattis, who oddly enough was actually a grad student at the time at Stanford, and he said, "You know what? If we're gonna teach computer science, when we first start out, why don't we have people not worry about all of the different commands of the language and all the different things they can do? Let's start with something really simple so you can learn all the commands real quick. And then you've mastered everything there is to master about that language, and you can focus on the software engineering concepts." And it turns out to be a brilliant idea, which has actually been adopted by a bunch of people.

And so Rich, who's a wonderfully friendly guy — sometime if we get him to come to Stanford, I'll introduce you; he's just very nice — came up with this thing called Karel the Robot. And the term, "Karel" actually comes from Karel Capek. Anyone know who he is? Oh, free candy. Uh huh?

Student: He coined the term, "robot."

Instructor (Mehran Sahami):He coined the term, "robot." He was a Czech playwright who actually wrote a play called, "RUR," which was about robots. And the word robot actually comes from a Czech word, the Czech word for work. And so the robot is named after Karel. And some people say Karl, which is kind of actually closer to I believe if — I don't know if there's anyone who speaks Czech in the room — but closer to the actual pronunciation. But we say Karel these days because it's kind of like gender neutral, okay?

And so Karel the Robot is basically this robot that lives in a really simple world. And so I'll show you all that you can meet Karel the Robot. He's friendly; he's fun. I'll show you Karel the Robot. So we gotta get Karel running. He's at the factory. He's getting souped up. We're energizing Karel. You gotta add some color to it. Otherwise — all right. We're begging for him. Come on, Karel. There he is. Oh, yeah. That's Karel the Robot. He looks like one of the old Macintoshes if you remember the original Macintoshes that look like a lunch pail, except he's got legs. One sticks out his back. That's just the way it is.

And the way Karel works is he lives in a grid. To you, it may not be exciting, but to Karel, it's way exciting. So Karel lives in this little grid, and the way the grid works is there are streets and avenues in the grid. Streets run horizontally, so this is First Street, Second Street, Third Street. And then over here, we have avenues, First Avenue, Second Avenue, Third Avenue, Fourth Avenue, Fifth Avenue. It's kind of like Karel lives in Manhattan if you wanna think about it that way, okay? So Karel always is on one of these corners. So right now, he's at the corner of First Street and First Avenue, or we just refer to it as 1 1 if you wanna think about sort of Cartesian coordinates, right? But just think of them as streets and avenues. That's where Karel lives.

And Karel can move around in this world. There's a bunch of things that Karel can do. He can take steps forward. He can turn around to face different directions, and he can sense certain things about his world. So there's some things that exist in Karel's world, okay? Things like walls that Karel cannot move through, right, so his world has walls all around it that he can't go through, so he can't fall off the end of the world. And there's other walls like this one if Karel were over here, he can't step through that wall.

There's also something referred to as beepers in Karel's world. And what a beeper is, is it's like a big diamond, okay? But what a beeper really is, is basically just some marker that he puts in the world. You can think of a beeper like a piece of candy. And Karel just goes around, like, putting pieces of candy in the world. As a matter of fact, not only does he put pieces of candy in the world, he carries around a whole bag of candy.

So he has a beeper bag with him, and sometimes that bag has a whole bunch of beepers in it; sometimes it only has one beeper; sometimes, it's sad Karel, and he has no beepers. But he's still got the bag. There just don't happen to be any beepers in it. So he can potentially, if he come across a beeper in his world, he can pick it up and put it in his bag, or he can take, if he's got beepers in his bag, he can take them out of his bag and put them places in the world. And corners in the world can have either zero — if they have no beepers, they just appear like a little dot — or one or more beepers on them that Karel can potentially pick up, okay?

So any questions about beepers or Karel having a little bag of beepers? And that's it. That's Karel. That's his world. His world, we can make it larger if we want. We can put in walls in different places. We can put beepers in different places. We can have Karel be in a different place.

But starting next time, what you're gonna realize is with this extremely simple world, there's actually some complicated things you can do. And after about a week — so this first week, we're gonna focus on Karel — you'll notice that Karel is actually a very nice, gentle introduction into Java. And a lot of the concepts that we learn, sort of software engineering concepts using Karel, will translate over to the Java world, okay? So any questions about Karel or any of the other logistics that you've actually heard about in the class?

Alrighty then. Welcome to 106A. I'll see you on Wednesday.

[End of Audio]

Duration: 50 minutes