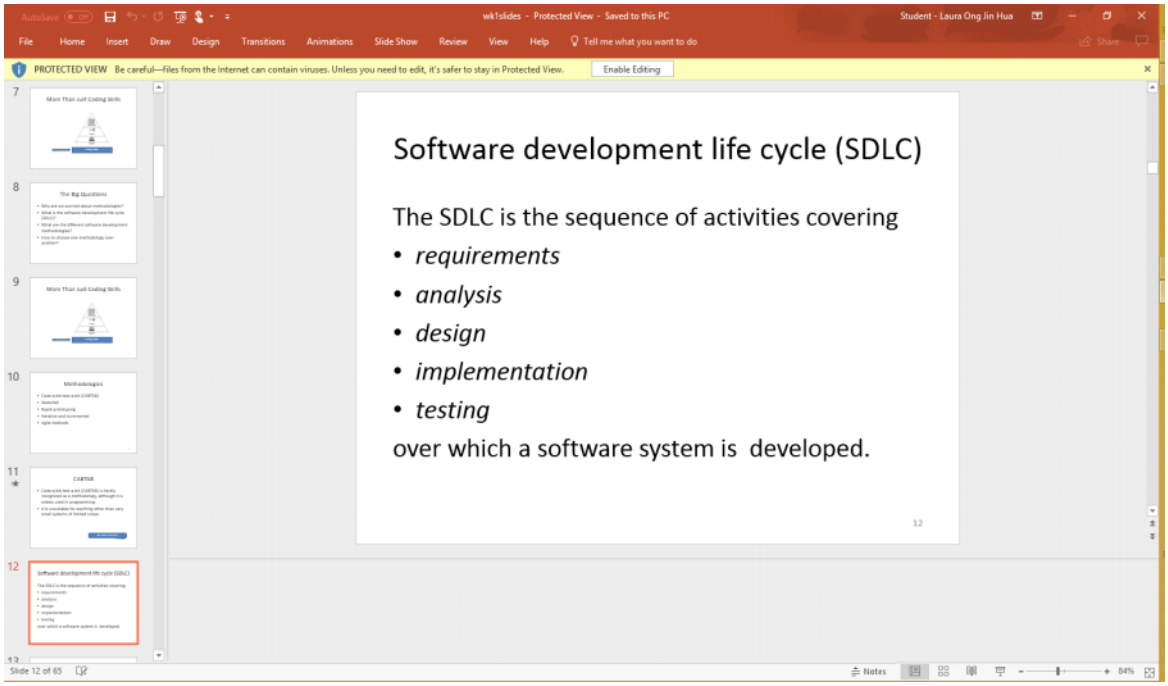
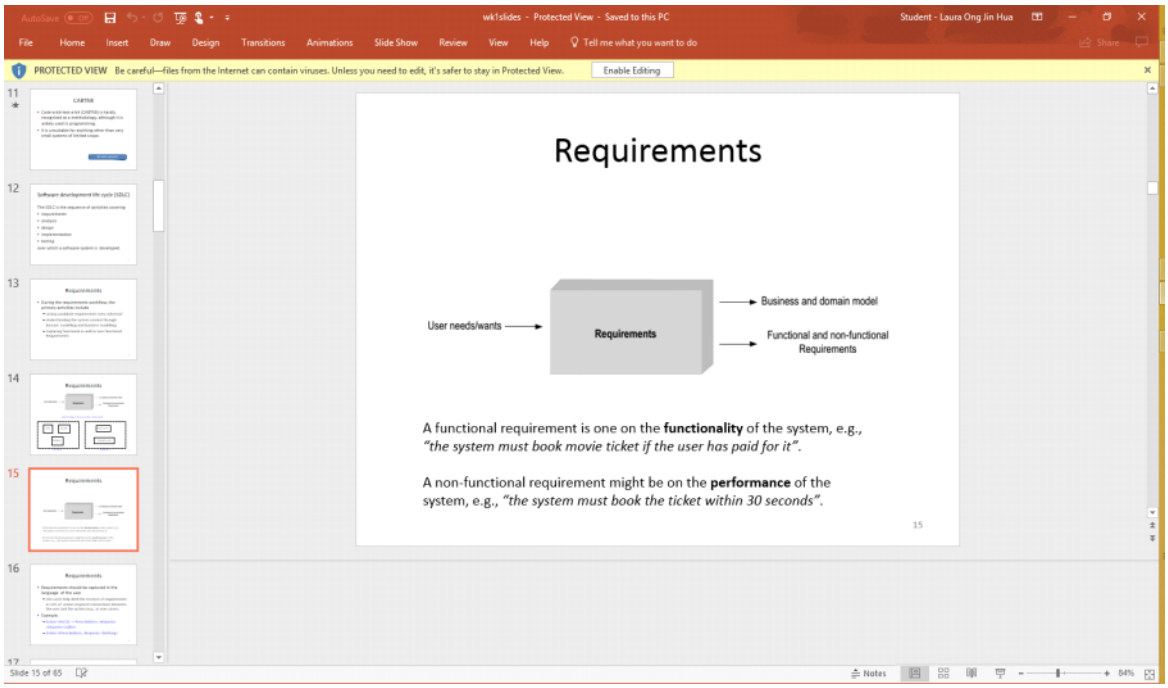
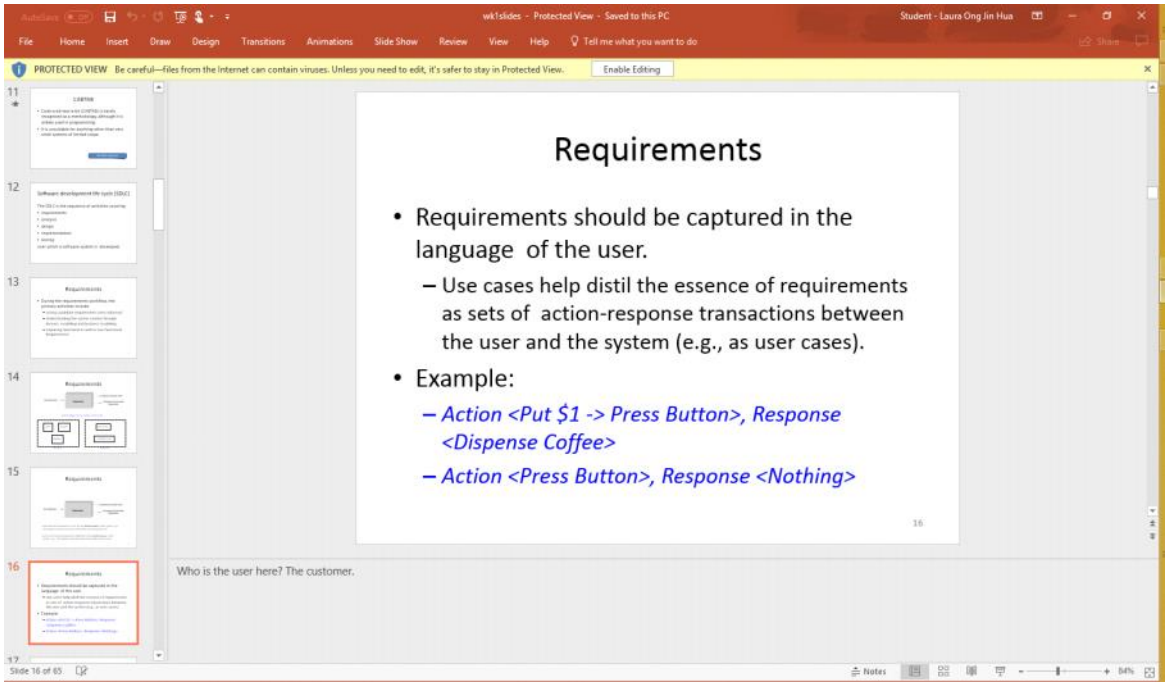
Cohort 1

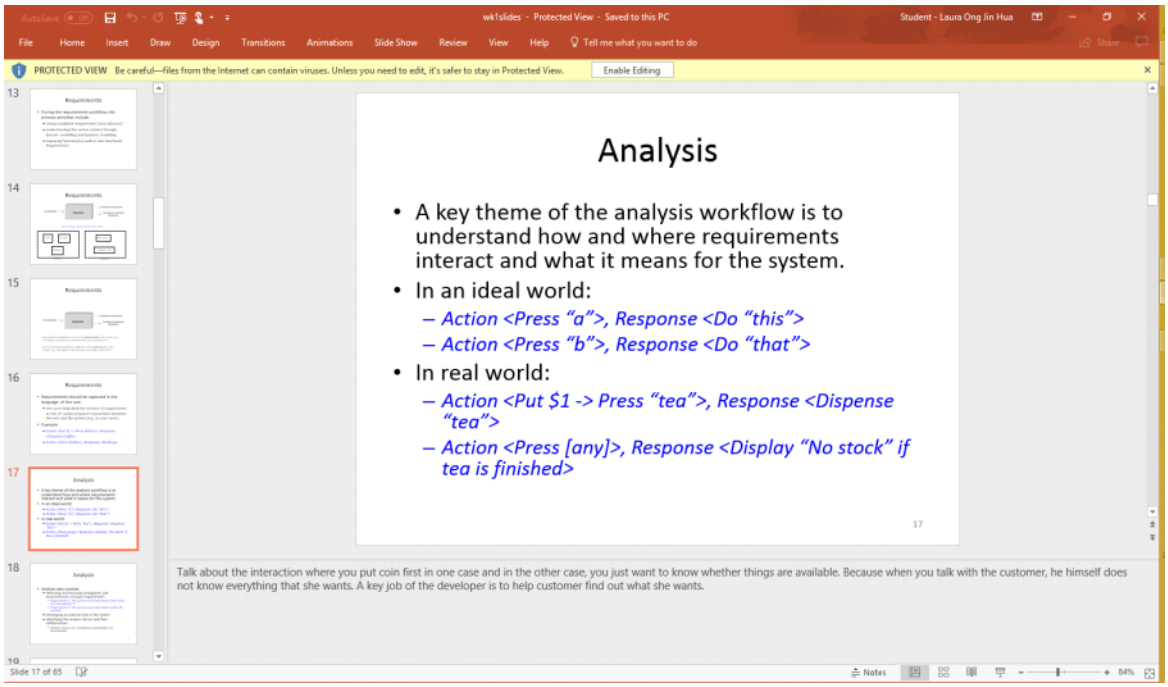
Monday, January 22, 2018 12:11 PM

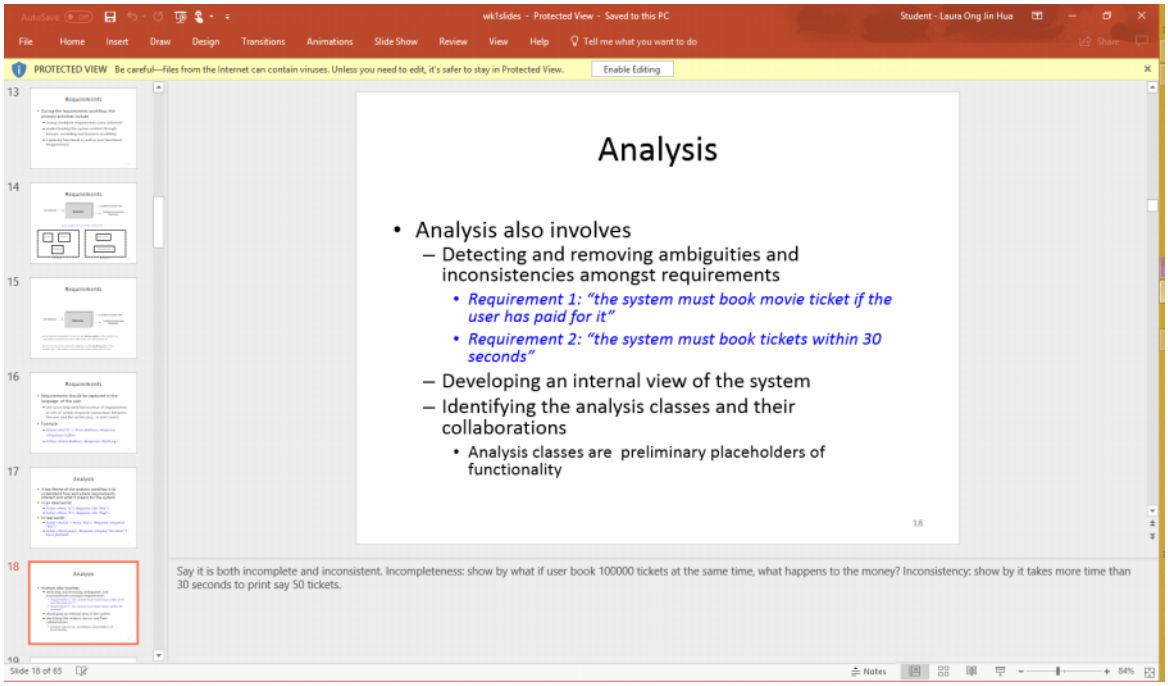


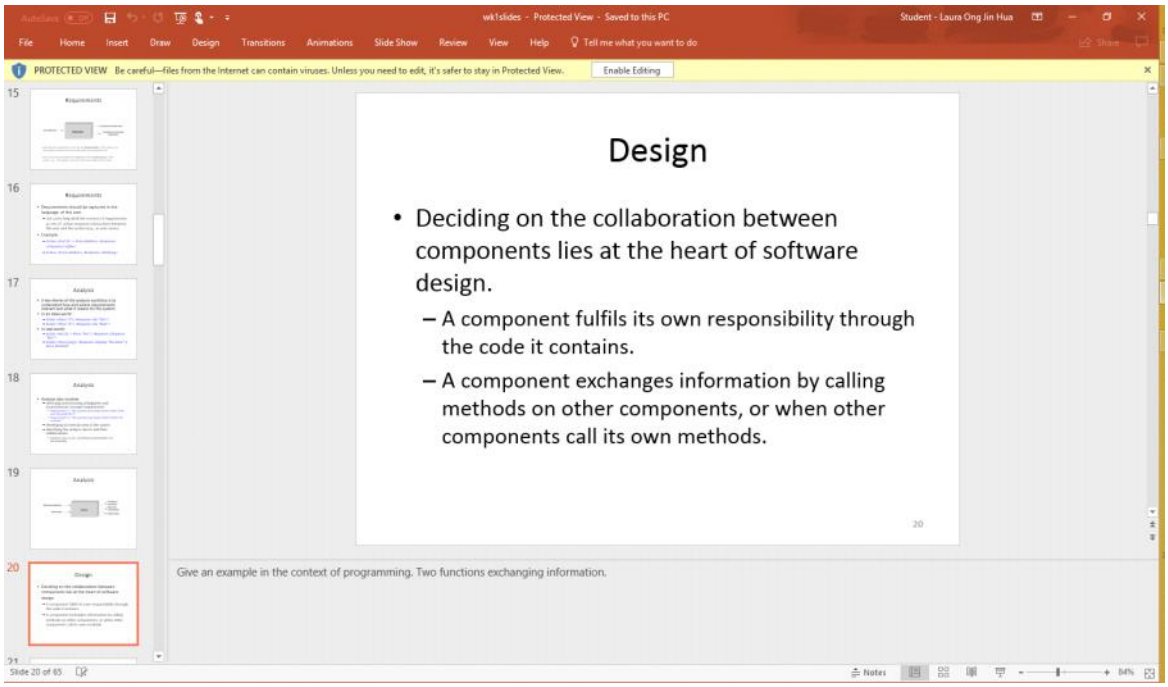
There are four kinds of methodologies:
Code a bit test a bit
Waterfall
Rapid prototyping
Iterative and inremental
Agile methods

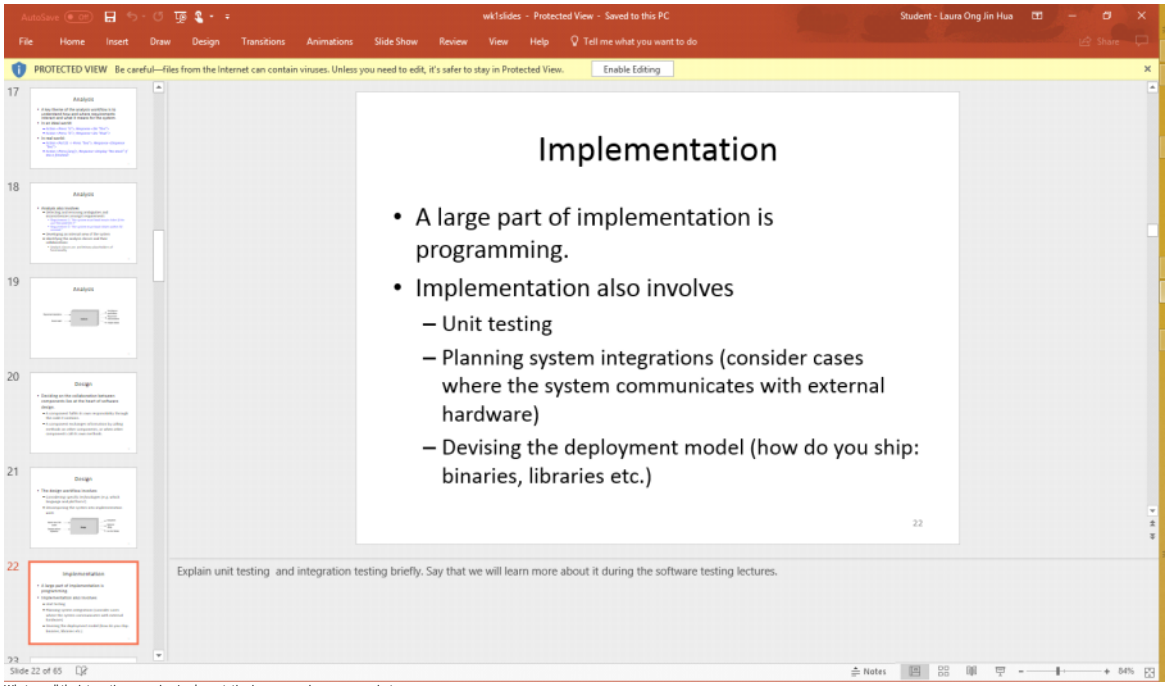



wk1slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
Requirements
• Requirements should be captured in the language of the user.
– Use cases help distil the essence of requirements as sets of action-response transactions between the user and the system (e.g., as user cases).
• Example:
– Action <Put $1 -> Press Button>, Response <Dispense Coffee>
– Action <Press Button>, Response <Nothing>
16
Who is the user here? The customer.
Slide 16 of 65
Notes


wk1slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
Analysis
• A key theme of the analysis workflow is to understand how and where requirements interact and what it means for the system.
• In an ideal world:
– Action <Press "a">, Response <Do "this">
– Action <Press "b">, Response <Do "that">
• In real world:
– Action <Put $1 -> Press "tea">, Response <Dispense "tea">
– Action <Press [any]>, Response <Display "No stock" if tea is finished>
17
Talk about the interaction where you put coin first in one case and in the other case, you just want to know whether things are available. Because when you talk with the customer, he himself does not know everything that she wants. A key job of the developer is to help customer find out what she wants.
Slide 17 of 65
Notes

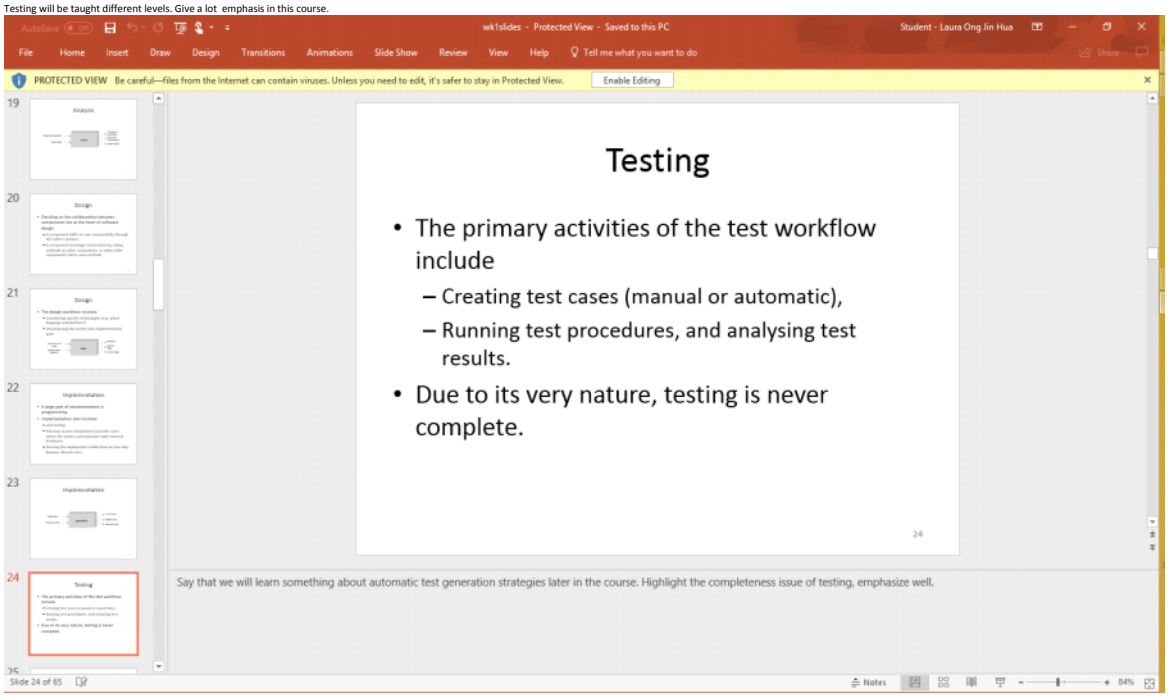

wk1slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
Analysis
• Analysis also involves
– Detecting and removing ambiguities and inconsistencies amongst requirements
• Requirement 1: "the system must book movie ticket if the user has paid for it"
• Requirement 2: "the system must book tickets within 30 seconds"
– Developing an internal view of the system
– Identifying the analysis classes and their collaborations
• Analysis classes are preliminary placeholders of functionality
18
Say it is both incomplete and inconsistent. Incompleteness: show by what if user book 100000 tickets at the same time, what happens to the money? Inconsistency: show by it takes more time than 30 seconds to print say 50 tickets.
Slide 18 of 65
Notes
84%

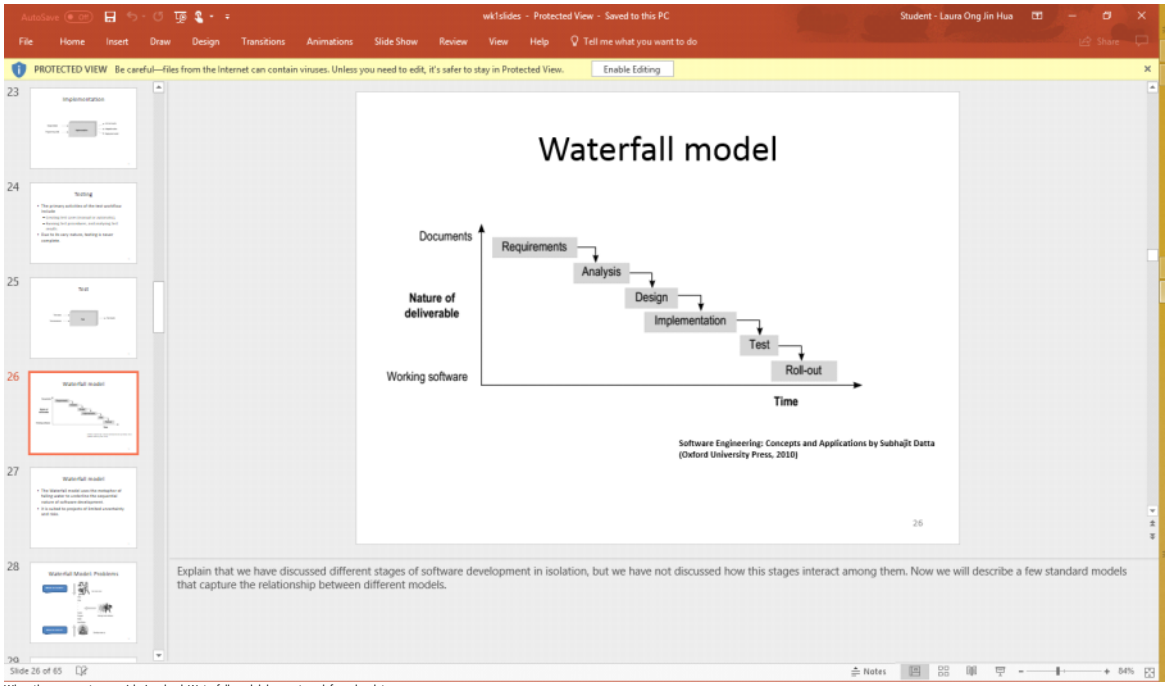

wk1slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
Design
• Deciding on the collaboration between components lies at the heart of software design.
– A component fulfils its own responsibility through the code it contains.
– A component exchanges information by calling methods on other components, or when other components call its own methods.
20
Give an example in the context of programming. Two functions exchanging information.
Slide 20 of 65
Notes
84%



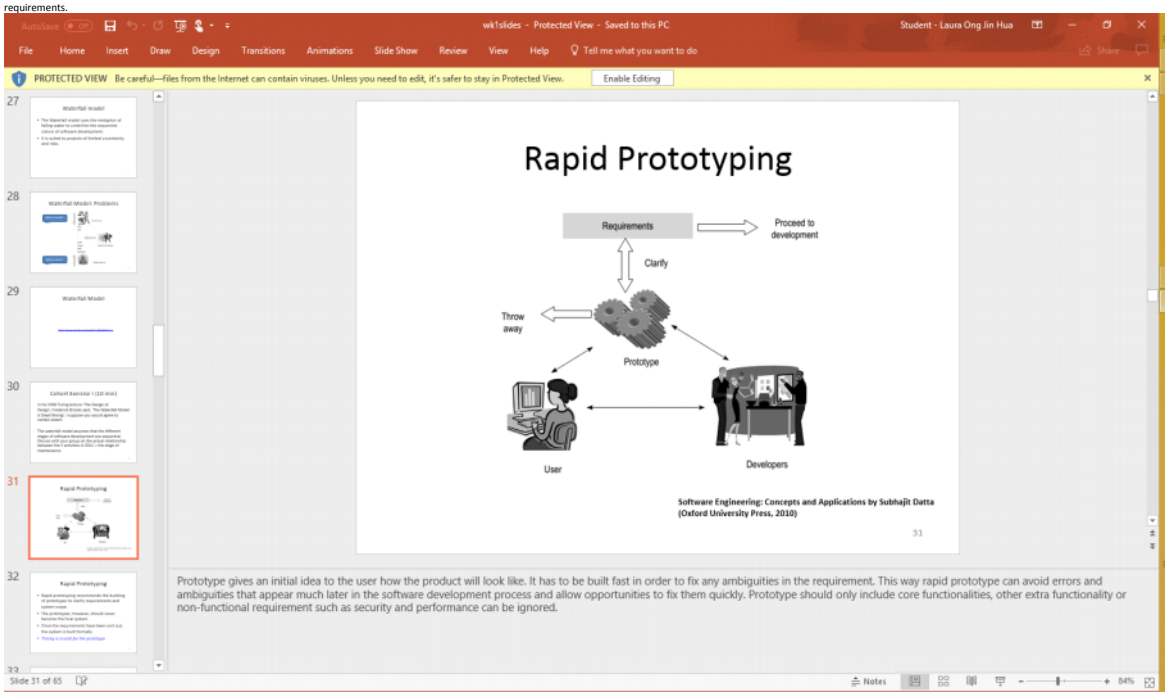
What are all the interaction scenarios. Implementation is programming on your project.

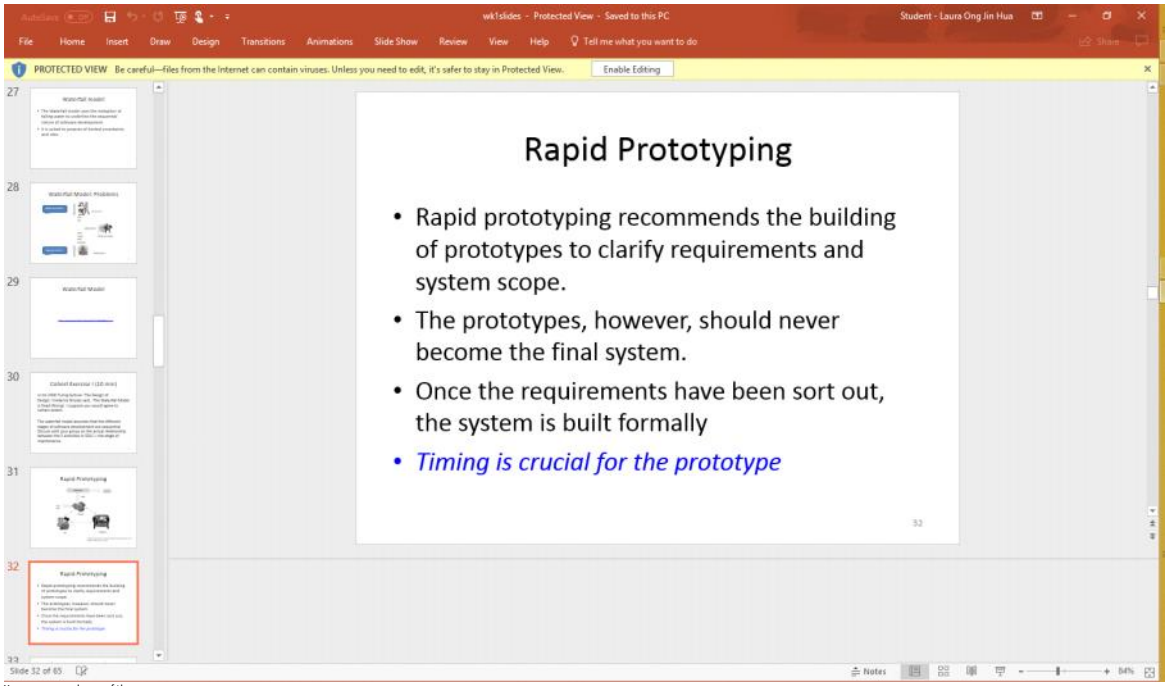
Testing will be taught different levels. Give a lot emphasis in this course.



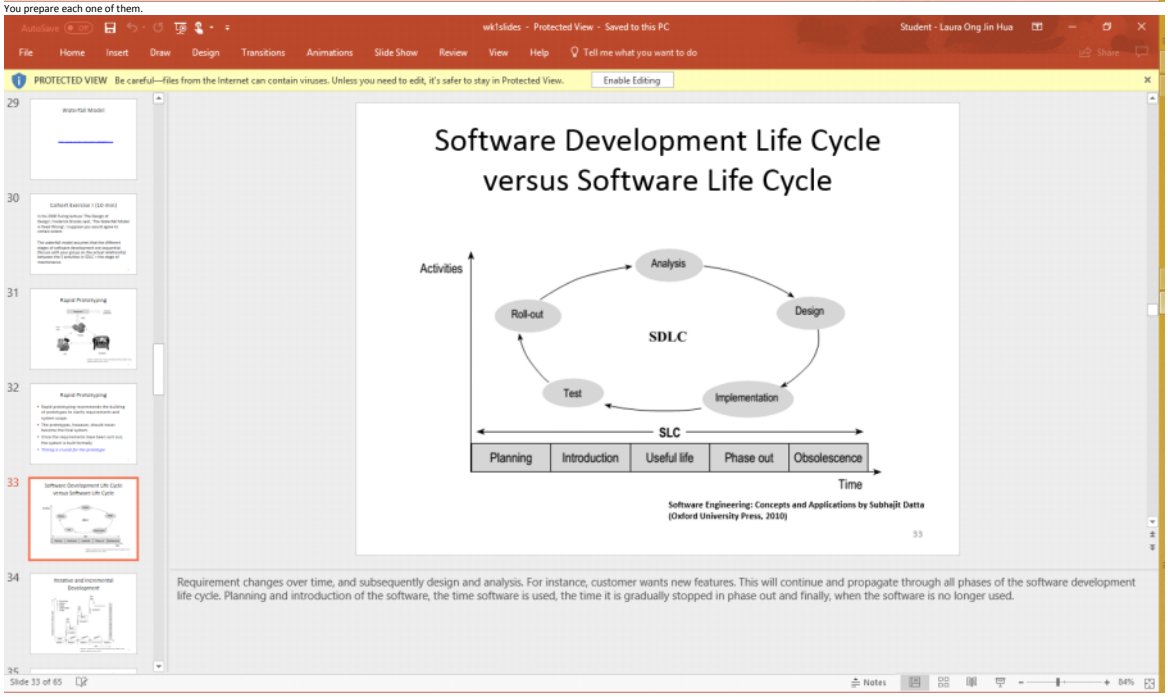


When there are not many risks involved. Waterfall model does not work for cyberdata.
Meant for limited risks and requirements. You do all the features, and once you have the prototype that you are happy with, then you do the final prototype that fulfils the requirements.





You prepare each one of them.



Cohort 2

Tuesday, January 23, 2018 11:34 AM

We talked about the different software development methodologies. Waterfall is a sequential order, and we saw that there were many problems there, because there could not be one-way dependency, need to revise as you go through the software development. One way is to use rapid prototyping, there is no sequential order of design implementation and testing, but make some prototypes to talk to the client. You go through with a few cycles and then you try with the software.

Iterative and Incremental Development. As you see this RADIT, these different states of software development. Let's say your project goes for 12 months, and timebox your project like two months, two months. You have this time box manner, and then you promise something and complete this set of features, and then you implement and put more features in the software. Similarly, you do some more features, and come up with an even greater software. It is very much like rapid prototyping, here you are not really making prototypes.

Rapid prototyping, you make something quick, you throw out the software and then make another. Here you are making the new software, then you put some more features, and then you have another release. So this is one way you follow your course project, because you can promise something.

You make different implements of the software, different versions, and you have different features being implemented.

Agile Manifesto: Agile is something which is not a software development methodology. A belief that people follow when they develop something. There are some diagrams, tools that people follow. Some tools that people follow when they use this software. Something which is more important and encouraged to follow. You use individuals and interactions. Your peers and your instructors and even TA. Working software over Comprehensive documentation: you show the clients what you have already done. This is a way to better communicate. It's the belief in the agile manifesto. You show that you communicate with the client. When you have something working, client feels better. Customer collaboration over Contract negotiation: you do not really collaborate with the customer with the project. You need to talk to clarify what you want in the project.

Responding to change over following a plan. Some people are not acceptable to making changes. You are trying to find out, that you have this additional feature as well. It's sometimes very common to be defensive about it. I'm not going to implement if he did not talk about it. Now that he or she has some working software, you might see that some additional features may be useful for us. When you design software, collaborate with the client, maybe he wants some features differently, the agile manifesto says that you should collaborate with it. Within the process, you could put a different software methodology.

Customer satisfaction by rapid delivery of useful software: We ask you to show something from the project. To convince you are making progress: instead of showing documentation, or some report, it's good to provide some software.
Welcome changing requirements. Let's say somebody tell you it want to be different; change very late, you should be welcoming it.
Working software is delivered frequently: usually working software is delivered infrequently, in one year. Agile has to be done in a few weeks. Working software is the principal measure of progress. When you have the project meeting, working software would make your grades better. Try to make this process so that it is better for us. Is this agile thing clear? It's not a development methodology. We can follow agile things simple.

Agile is mostly about timeboxing. It is possible it might be this time but different. Say your software has 10 different features. Let's say you have 2 features first, then 4 then 4 last features. Agile says that you should do in timeboxes. Instead of completing certain sets, you will try to implement two features. Possible to implement something more. We have provided some requirement for every project meeting. But it doesn't stop you from going beyond. Meeting every fixed set of time rather than fixed set of principles.

There are things on which your project depends. Some empirical data.
(b^1/3 * Size)/Productivity = Effort ^1/3 * time^4/3
B = special skill factor
Productivity = a productivity parameter (depends on team)
Effort = Human effort in years or months
Time = time to complete project in years or months.

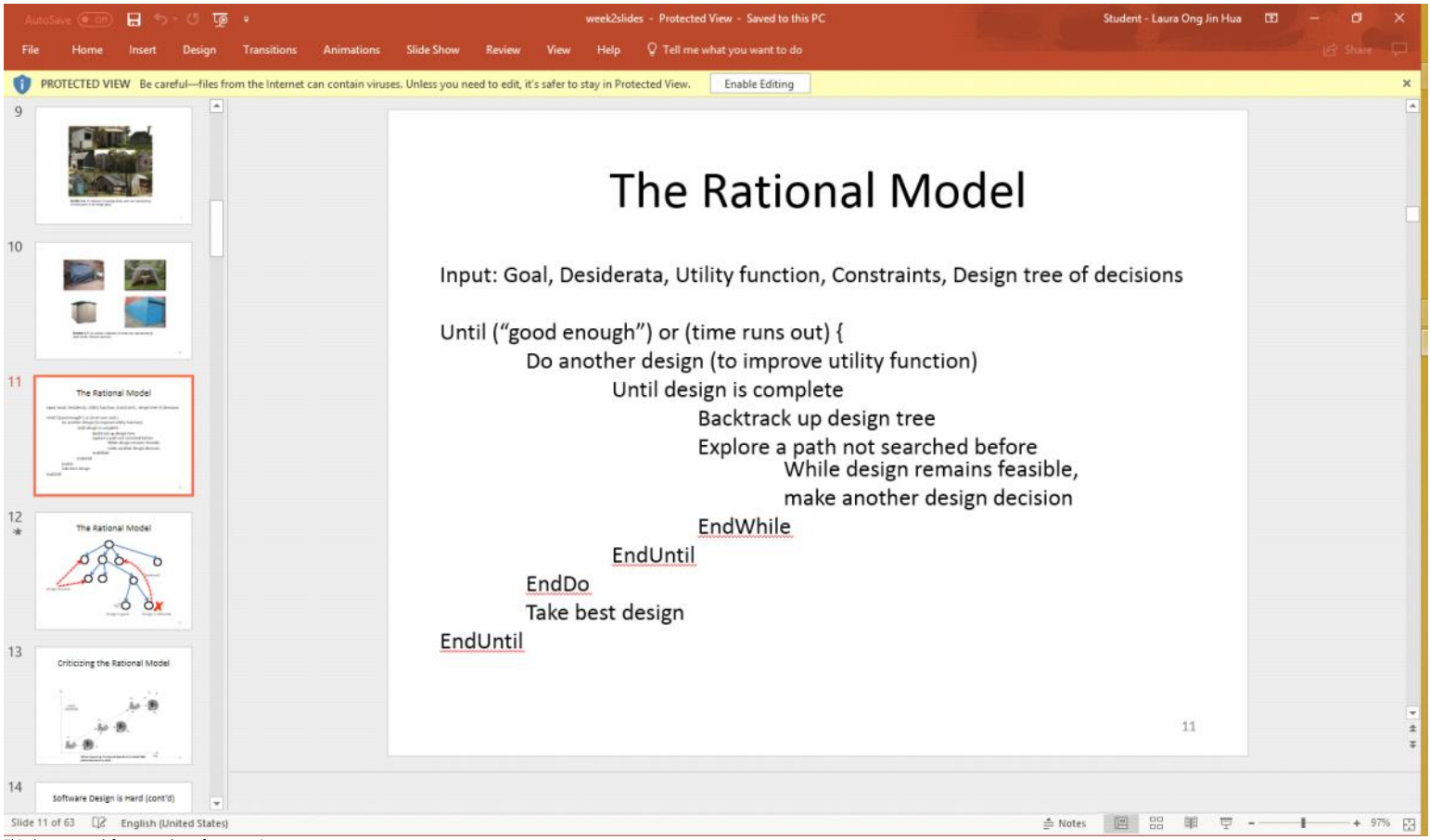
Given that three months time, how much effort you intend ot put.
Effort is the number of willing hours to put into your project. If you reduce the time, the time needed to reduce the projext reduced by 1 but increased by a factor of 4.
What is the implication? Any schedule slip, say if your group does not work for a few amount of time, it increases the effort a lot more than these few impact hours. Say you did not work for two days, It does not mean 8 days effort would be required. B is some constant. Now what I want to show you it's that it is a power of four.

B is related to the size of the product. More code you develop, more skill you need.
5-15K = 0.16

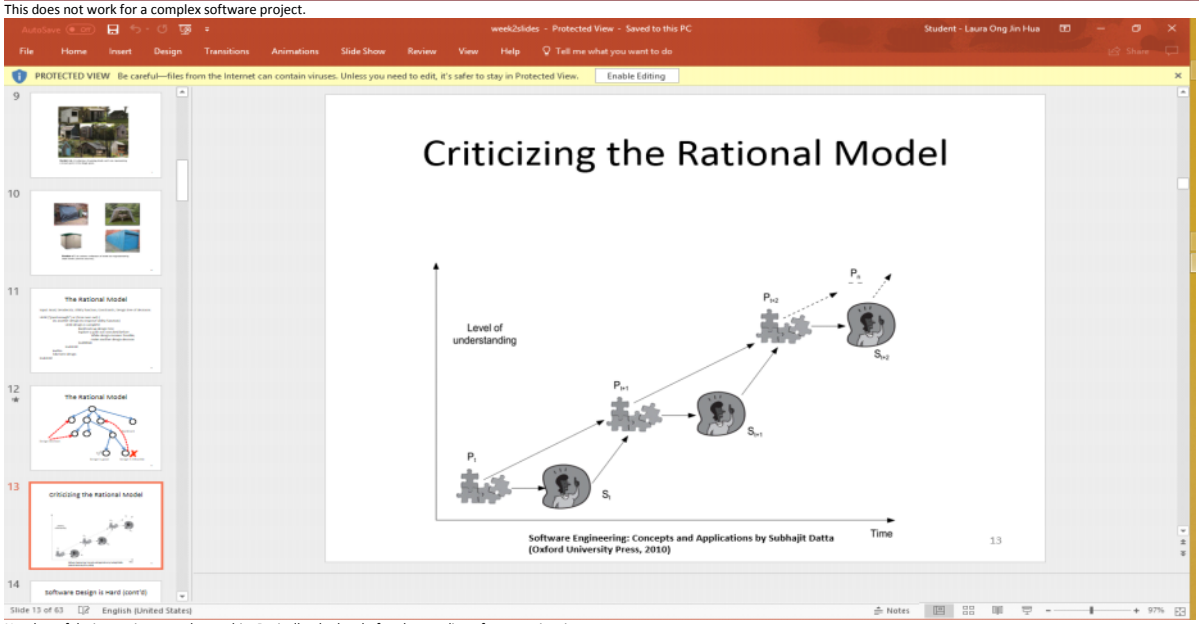
Productivity is the type of software being development.

Introduction to Software Design

Monday, January 29, 2018 11:30 AM



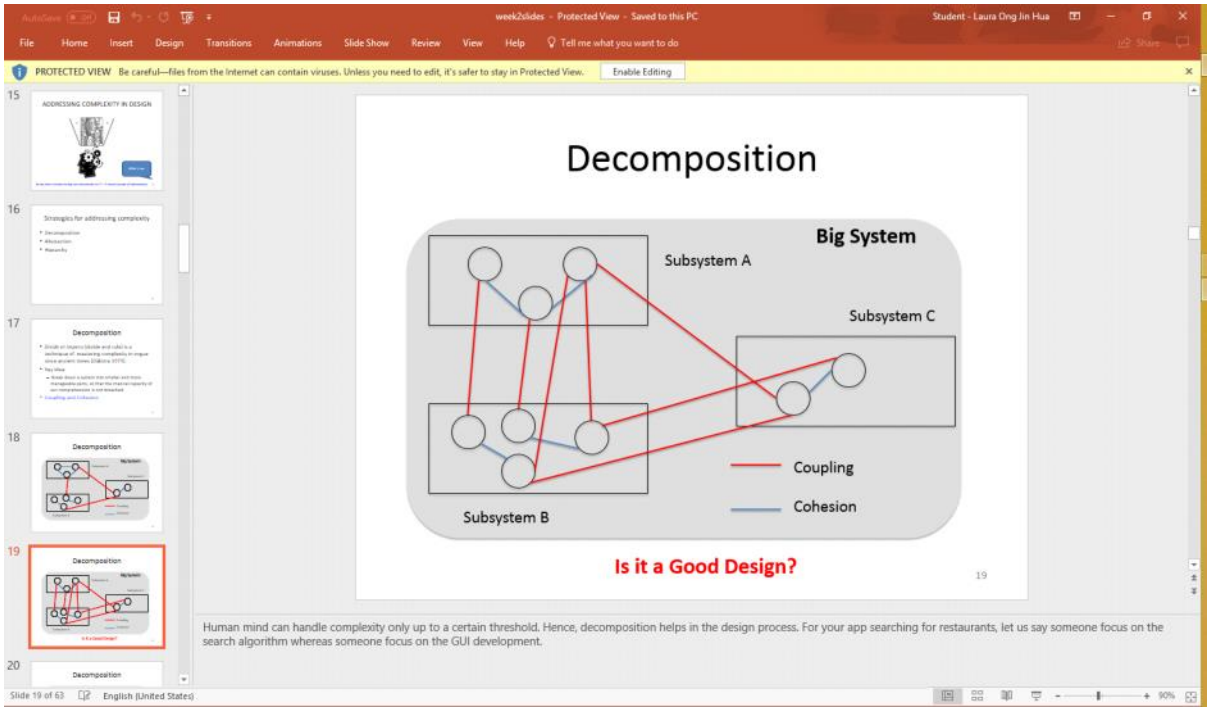
This does not work for a complex software project.



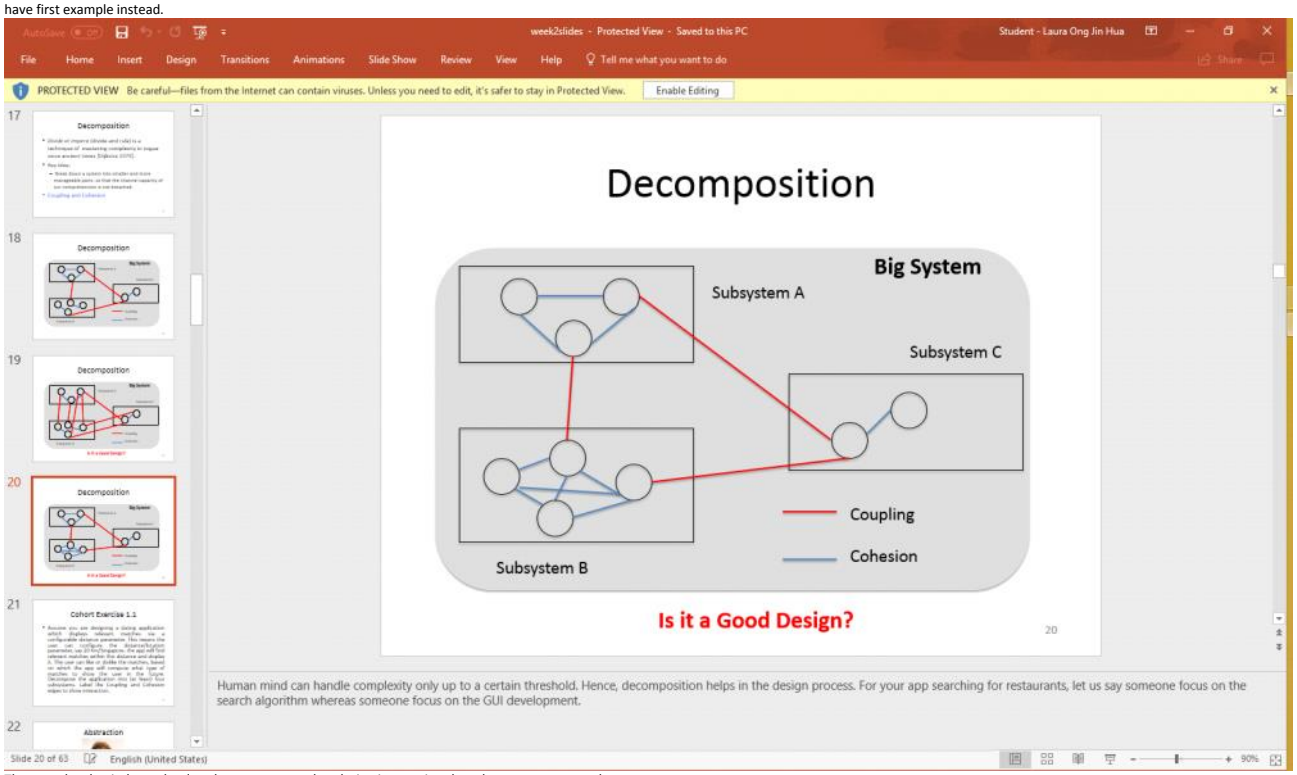
Number of design options may be too big. Basically, the level of understanding of your project increases with time. For any internal project, any options become better. Even if we know all the design choices, the combinations of design choices are exponential and therefore it may not be easy to find a good design. Enumeration does not work.

Strategies for addressing complexity:
Decomposition, Abstraction, and Hierarchy.
Decomposition: Divide and conquer. To solve harder problem, simplify by going to the smaller problem.
If I have a big software project, how can I decompose the software project into simpler software systems.

Break down a system into smaller and more manageable parts, so that the channel capacity of our comprehension is not breached.



Here, not a good design. Better to combine sub systems together to minimize complications. Better to have first example instead.



They can develop independently, when merge together during integration then they can come together.

Dating Application
: Find filters (for dating partner)
:Take the addresses from the filtered results and cross examine with the results
:Takes the user's input
:Use the input to re-filter results
:UI
:Database
:Algorithm (For recognition)
:Calculation
Coupling is between input to find filters, addresses from filters to find relevant partners. Within the Cohesion edges, find the inputs as to determine the likeliness/probability of getting the thing.

week2slides - Protected View - Saved to this PC — Student - Laura Ong Jin Hua

File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

21 Cohort Exercise 1.1

22 Abstraction

23 Abstraction

24 Abstraction

25 Abstraction

26 Abstraction

Abstraction

The power of abstraction

Abstraction is at the center of much work in Computer Science. It encompasses finding the right interface for a system as well as finding an effective design for a system implementation.

22

Slide 22 of 63 English (United States) Notes 90%

week2slides - Protected View - Saved to this PC — Student - Laura Ong Jin Hua

File Home Insert Design Transitions Animations Slide Show Review View Help Tell me what you want to do

PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View. Enable Editing

21 Cohort Exercise 1.1

22 Abstraction

23 Abstraction

24 Abstraction

25 Abstraction

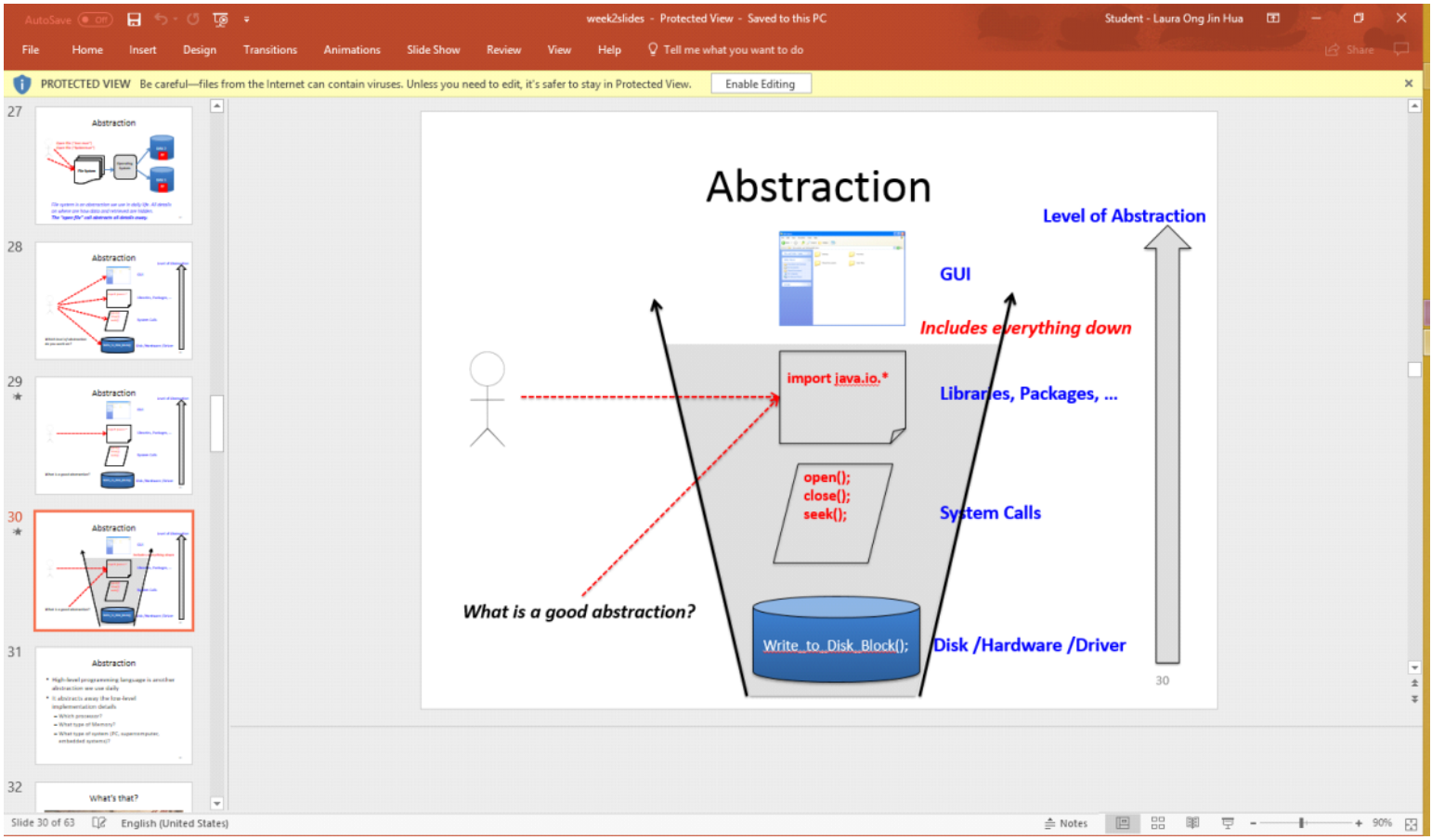
26 Abstraction

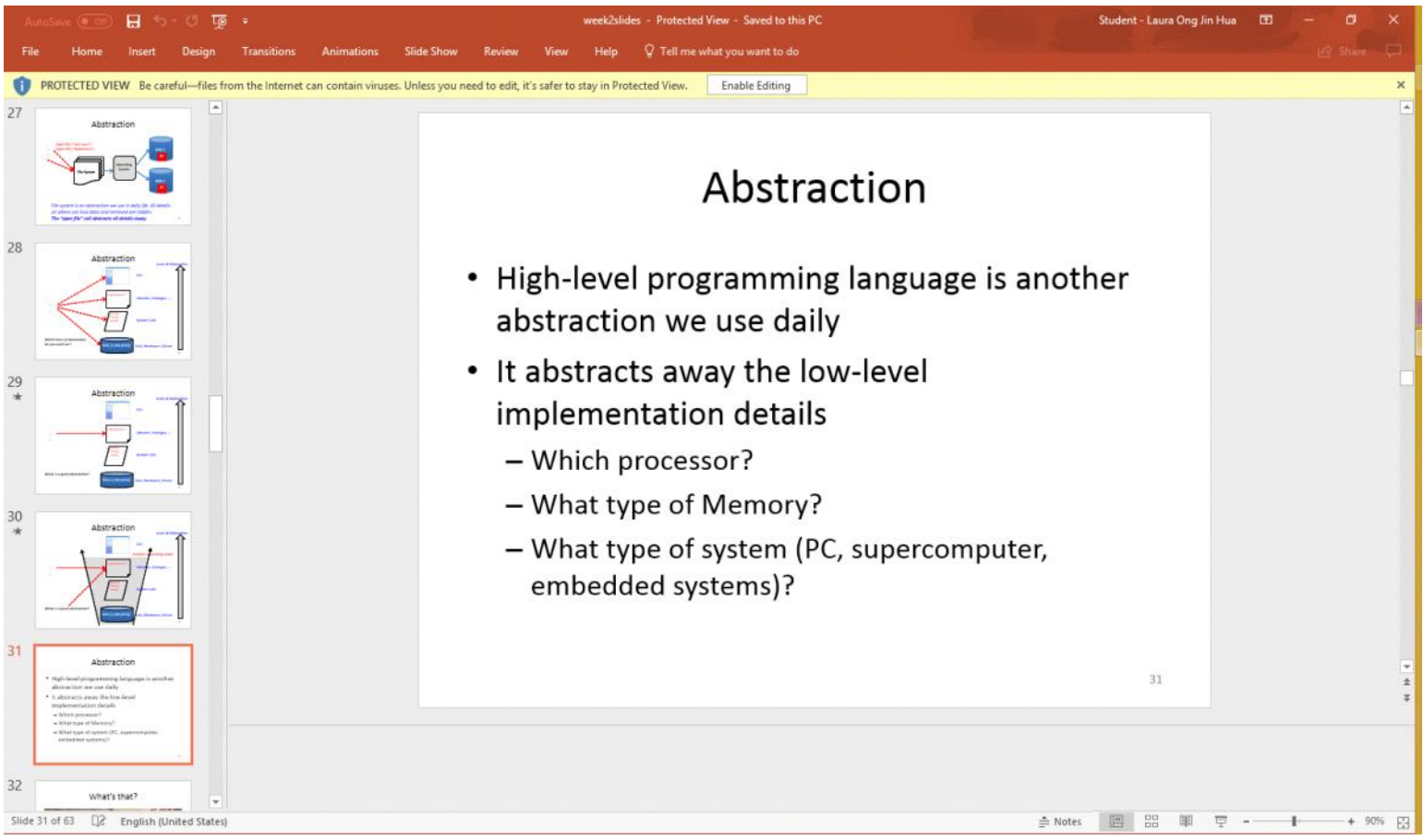
Abstraction

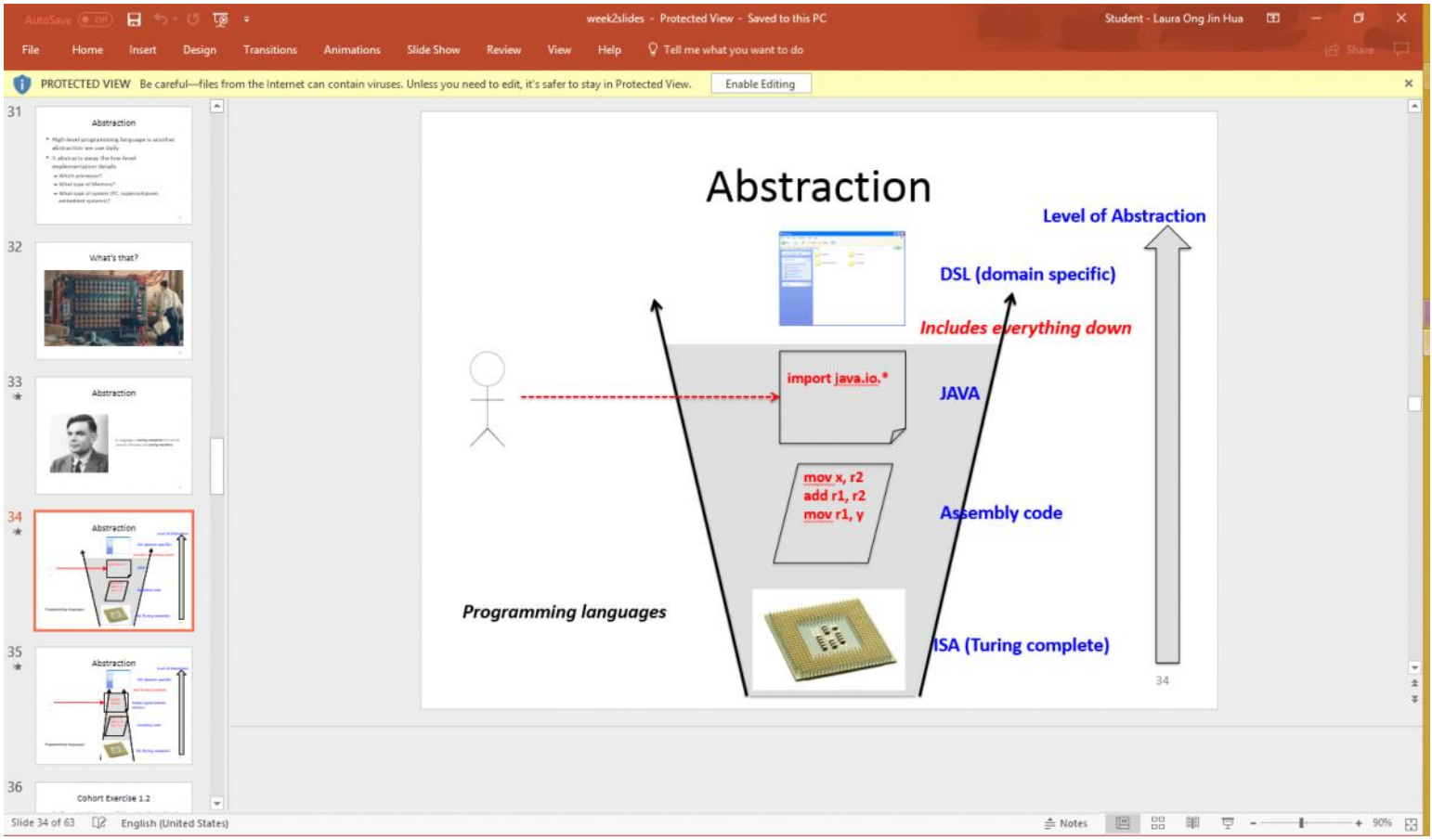
- One of the most frequently used words in the context of software design
- Key idea:
  - Unable to master the entirety of a complex object, we choose to ignore its inessential details, dealing instead with the generalized, idealized model of the object [Booch et al. 2007].

23

Slide 23 of 63 English (United States) Notes 90%

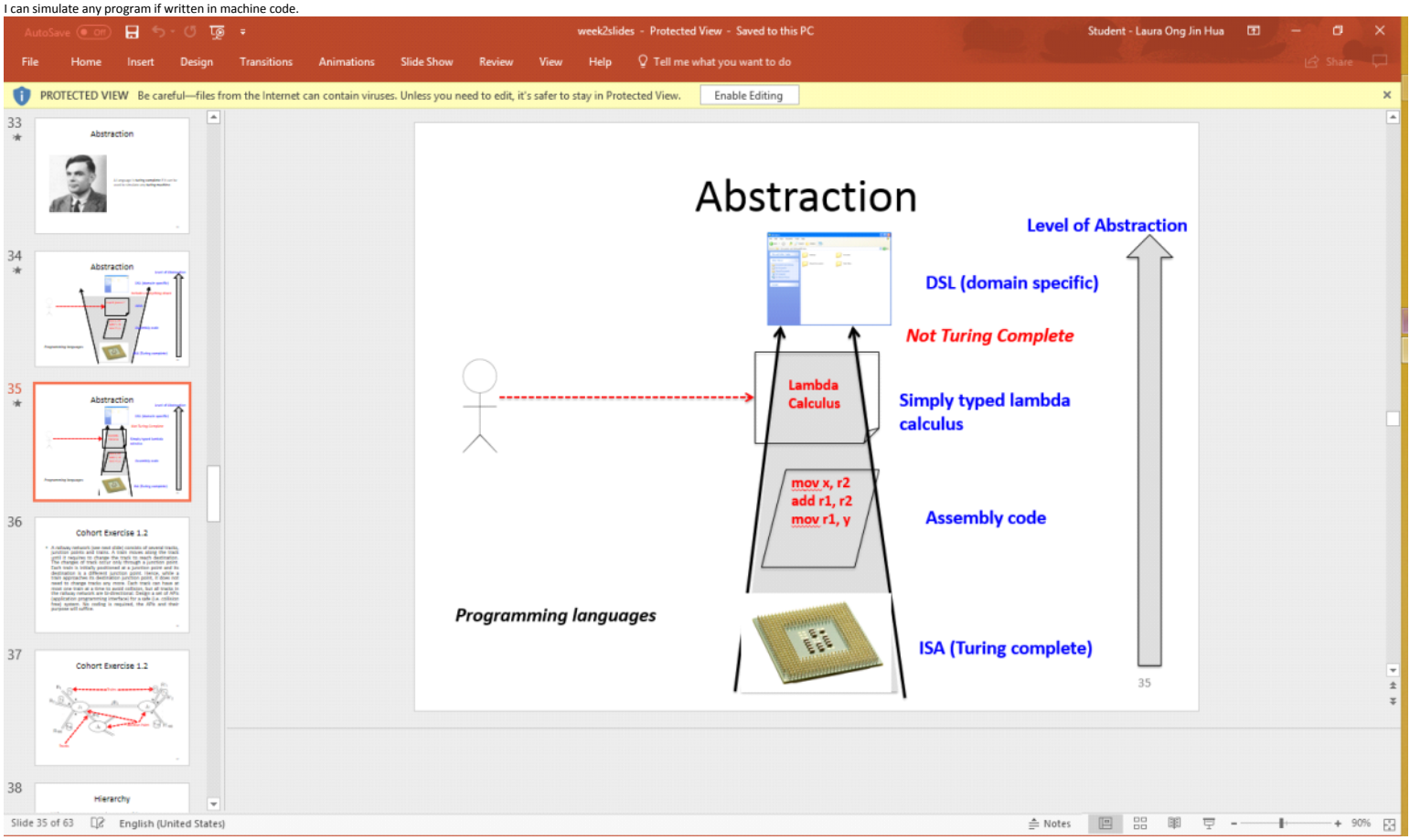

week2slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
Abstraction
Level of Abstraction
GUI
Includes everything down
import java.io.*
Libraries, Packages, ...
open();
close();
seek();
System Calls
What is a good abstraction?
Write_to_Disk_Block();
Disk /Hardware /Driver
30
Slide 30 of 63
English (United States)


week2slides - Protected View - Saved to this PC
Student - Laura Ong Jin Hua
PROTECTED VIEW Be careful—files from the Internet can contain viruses. Unless you need to edit, it's safer to stay in Protected View.
Enable Editing
Abstraction
• High-level programming language is another abstraction we use daily
• It abstracts away the low-level implementation details
– Which processor?
– What type of Memory?
– What type of system (PC, supercomputer, embedded systems)?
31
Slide 31 of 63
English (United States)



A Language is **turing complete** if it can be used to simulate any **turing machine**.

I can simulate any program if written in machine code.



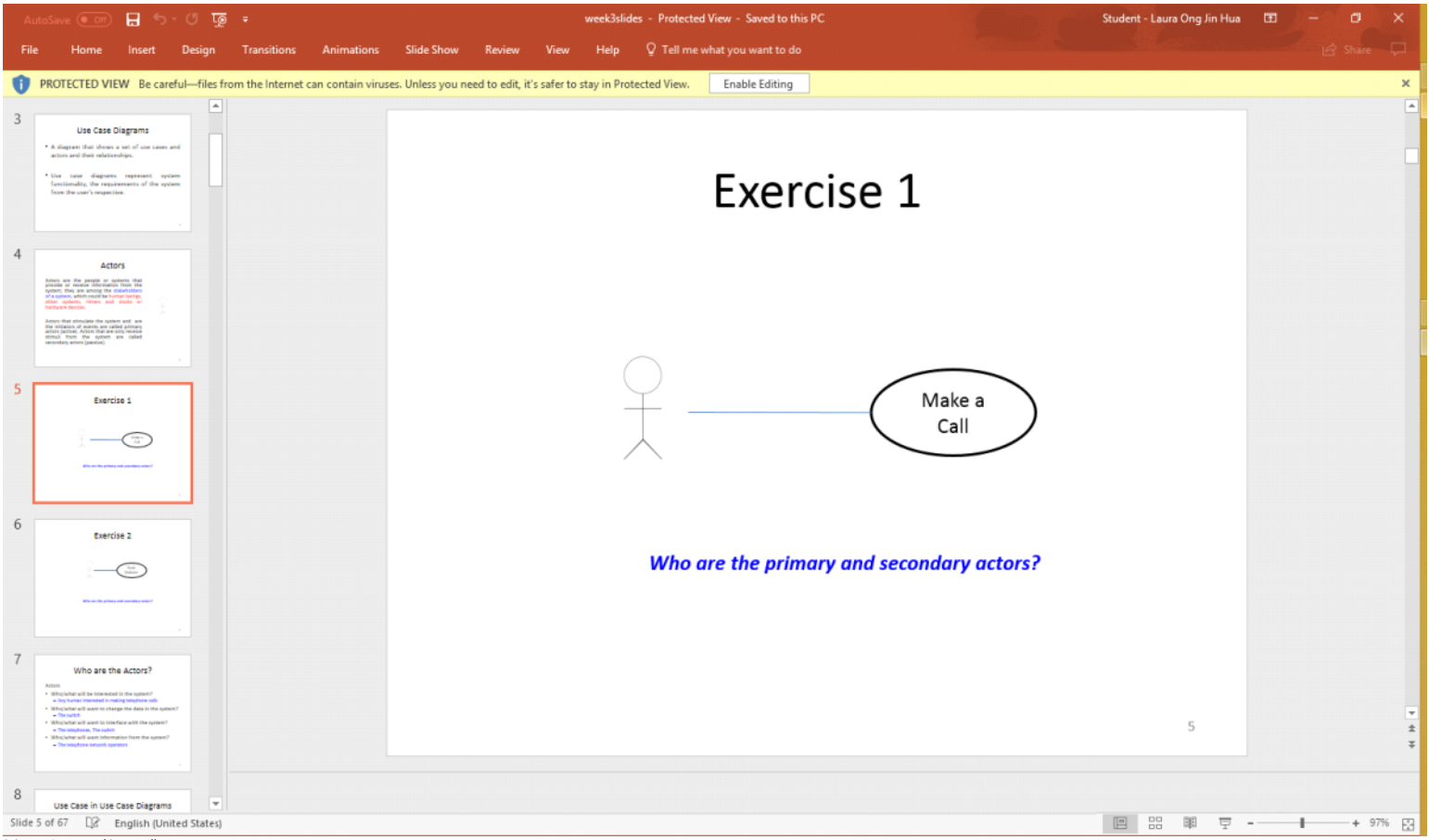
Something that can be done in machine level, will not be done in the lambda calculus. The concept of turing complete does not comply: i.e. HTML
Hierarchy: Whenever we examine something new or something that seems to be complex, we try to see

what common characteristic it shares with some other things we know, or if it is part of something we have seen earlier.
Key idea: "is a" relation: a car is a locomation device with wheels. Class Car extends locomotion
"Part of" relation: A part of a car is the engine. Engine e is part of Class c documentation.

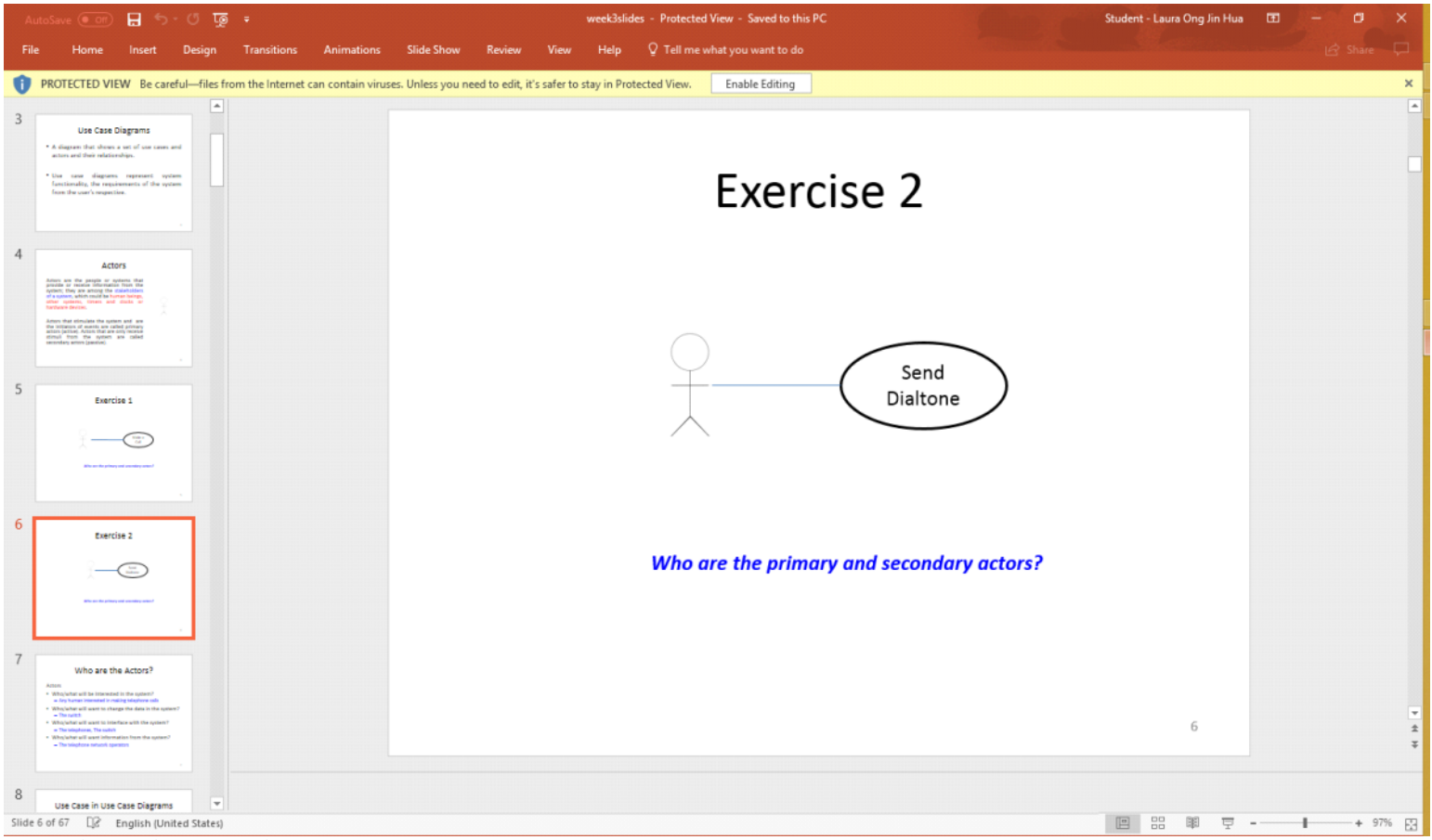
Use Case:

Monday

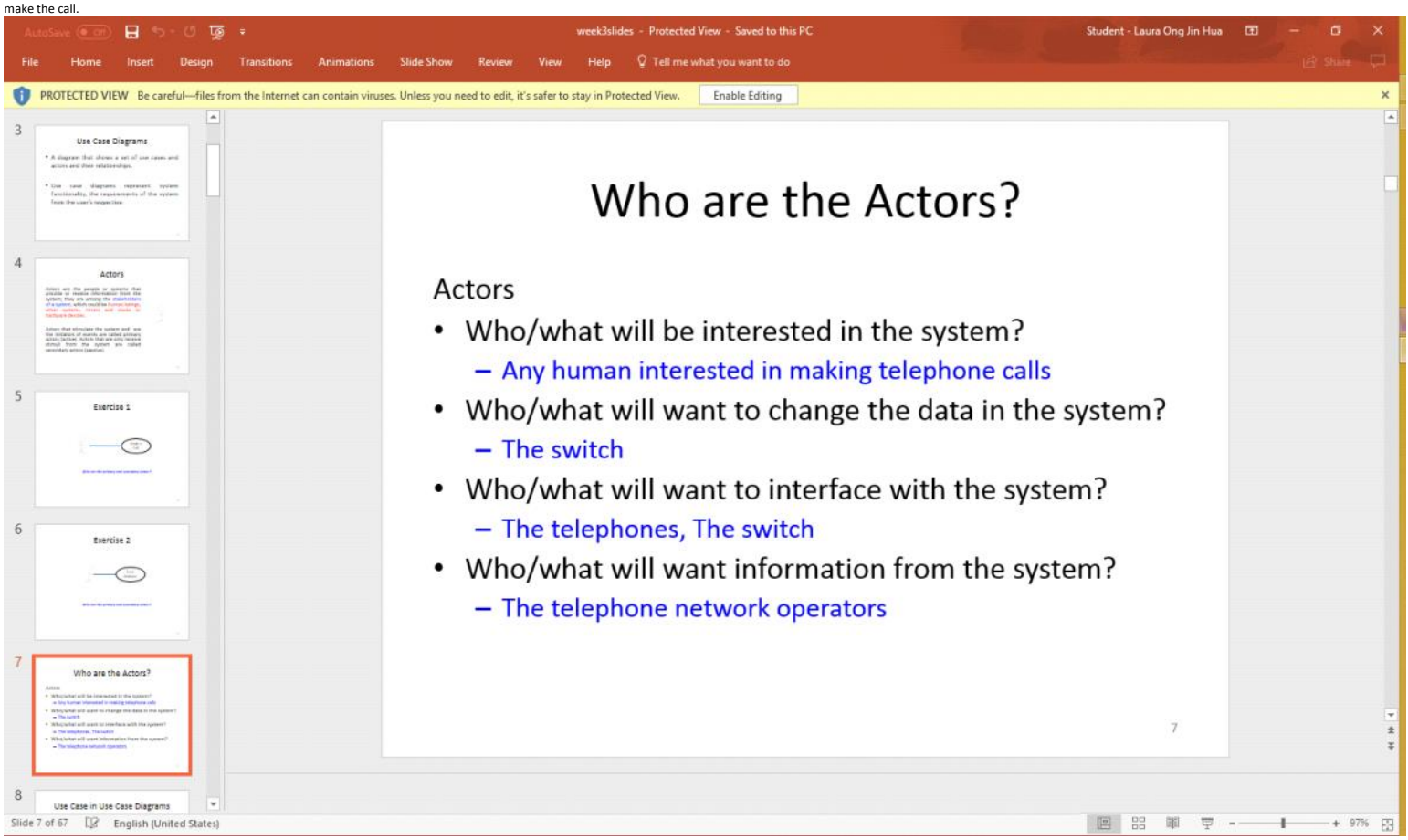
Monday, February 5, 2018 11:44 AM

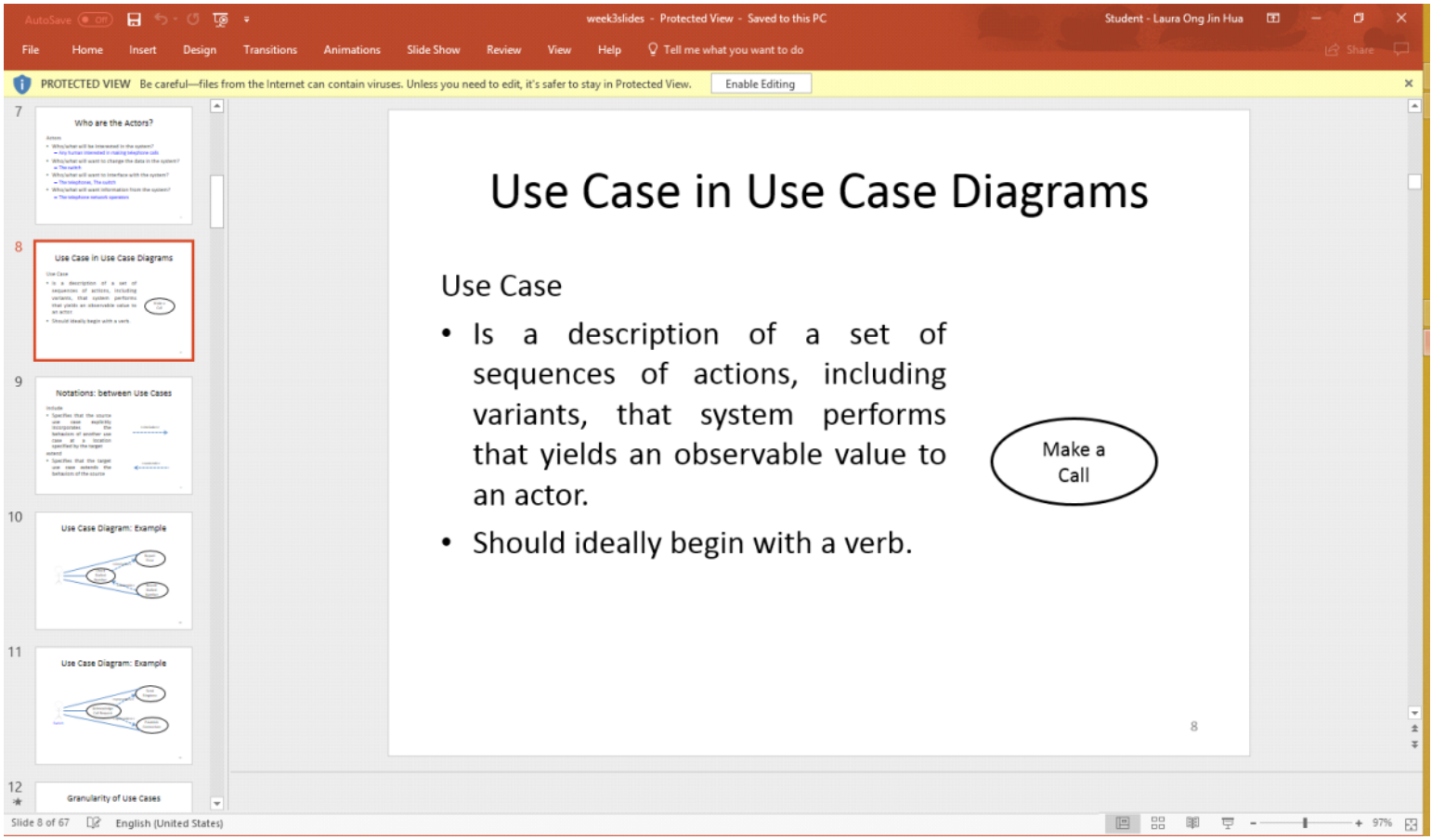


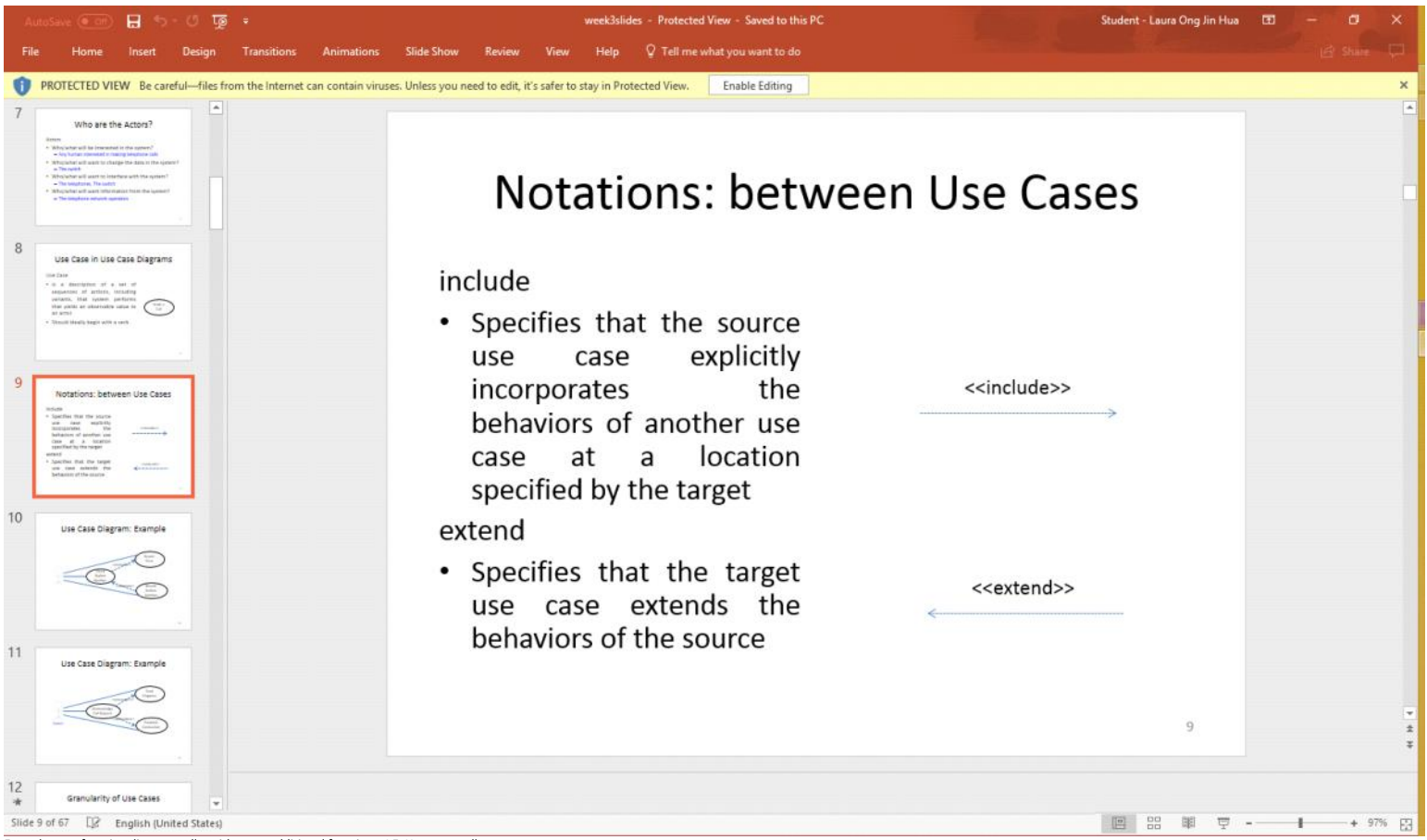
Primary Actor: making a call
Secondary Actor: switch operator



Actor: Switch (subsystem of telephone network). Secondary Actor is the subscriber, whoever is trying to make the call.







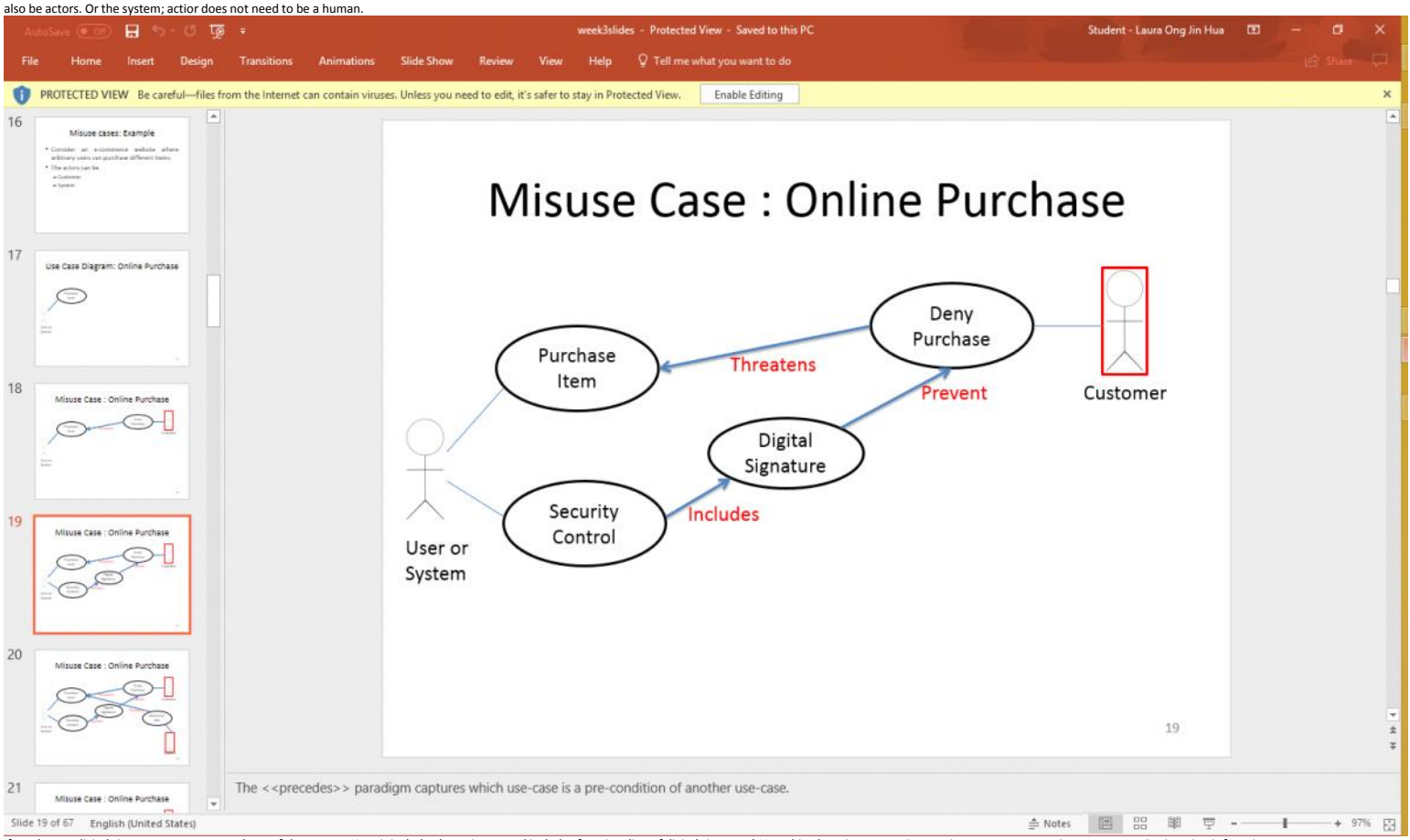
Extend: same functionality generally, with some additional functions. I.E. Have a enrollment course use case, and extends the email exchange student enrollment
Includes: library .i.e. error inclusion

Tuesday

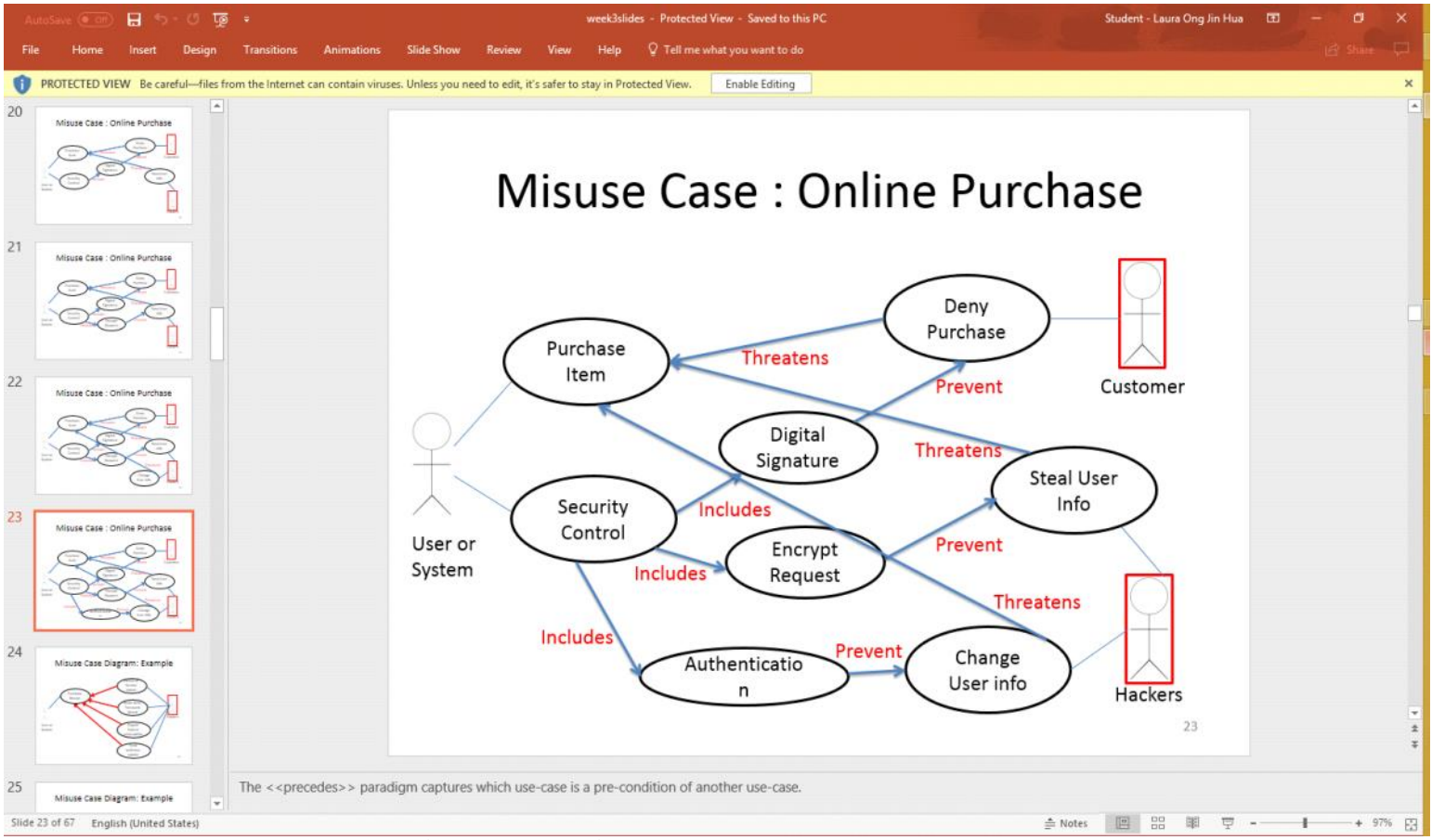
Tuesday, February 6, 2018 11:40 AM

Misuse cases: Not a good idea when you design a system to test only for the use cases. Must consider ways system can be abuse d. Quite related to the security or any kind of non function entity in general. If you are talking about the telephone networ k, only consider what you are supposed to do. There are ways the system can be abused. There might be some loop hole which someone ca n abuse. Hacking and network is one. Non functional requirements count as part of the misuse cases.

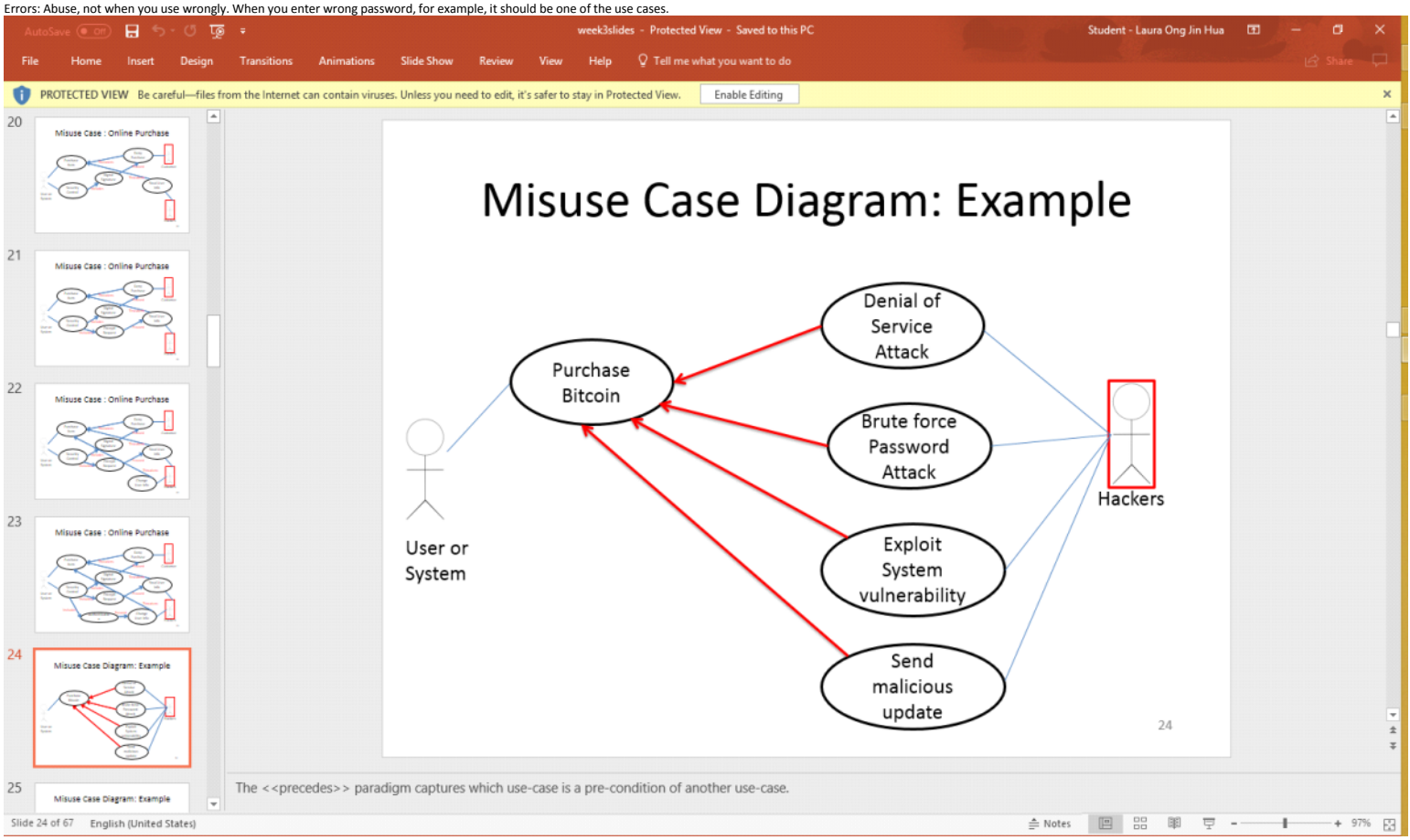
Client will not consider misuse cases. Just because the client don't give it to you does not mean it doesn't want to give to you. Give an example how complicated it can become even for a simple use case diagram. Actors can be the customers. Hackers c ould also be actors. Or the system; actior does not need to be a human.

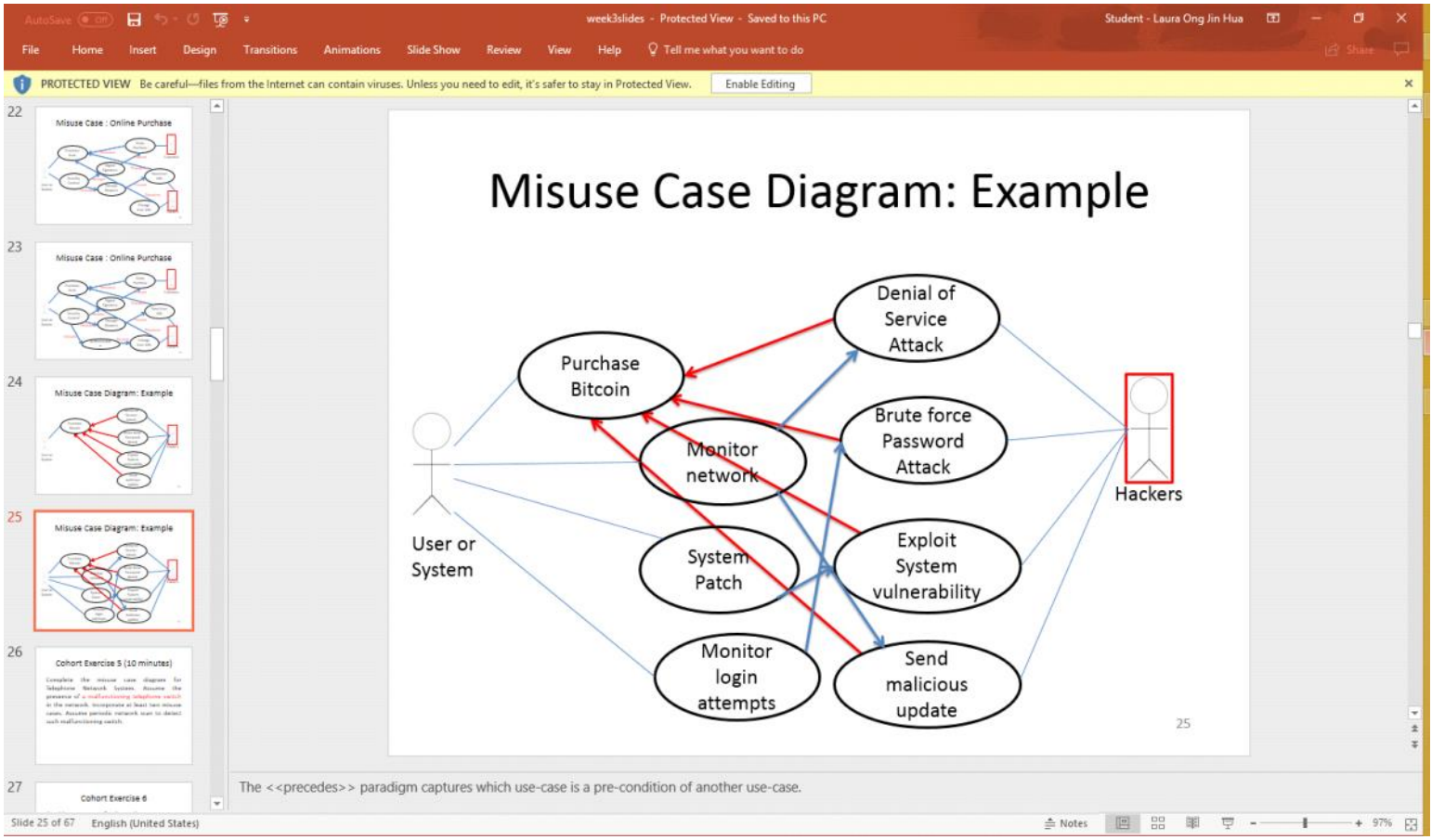


If you have a digital signature, can prevent abuse of the system. Here it includes (security control includes functionality o f digital signature) New thing here is prevent. Prevent is some component in your system. Designation is for misuse cases.

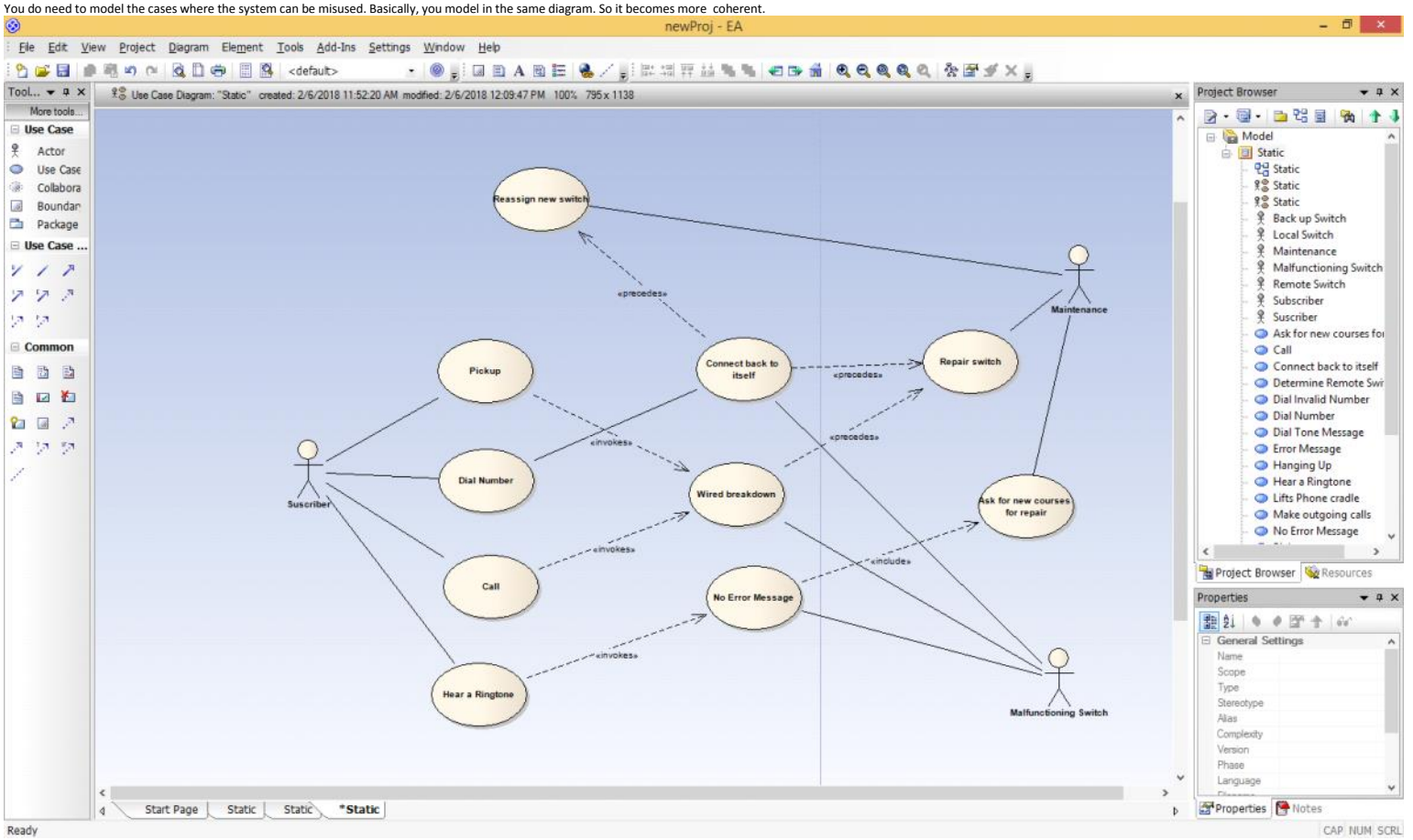


Errors: Abuse, not when you use wrongly. When you enter wrong password, for example, it should be one of the use cases.





You do need to model the cases where the system can be misused. Basically, you model in the same diagram. So it becomes more coherent.

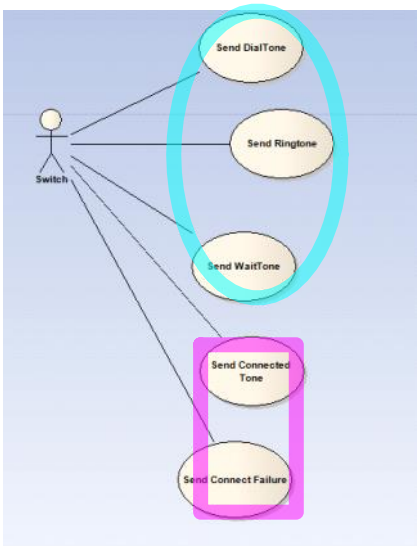


Class diagrams: What are the components of your system. You may not have come across a Class Diagram. IT is when you try to c onstruct something object oriented. It shows you the different classes of your software and the interactions between them.

Interfaces: dotted line. There should be a class that implements that interface.

Every use case for the switch has to be called by some operation. Every operation in the class has to correspond to some use case. Which means you have something redundant in your class. If you have some use case which is nothing in the class; you ar e not implementing system correctly. But it doesn't have to be one to one. At the same time, every function in the class has to cor respond to some use case. So everything becomes connected. Let's assume I am abstracting it away.

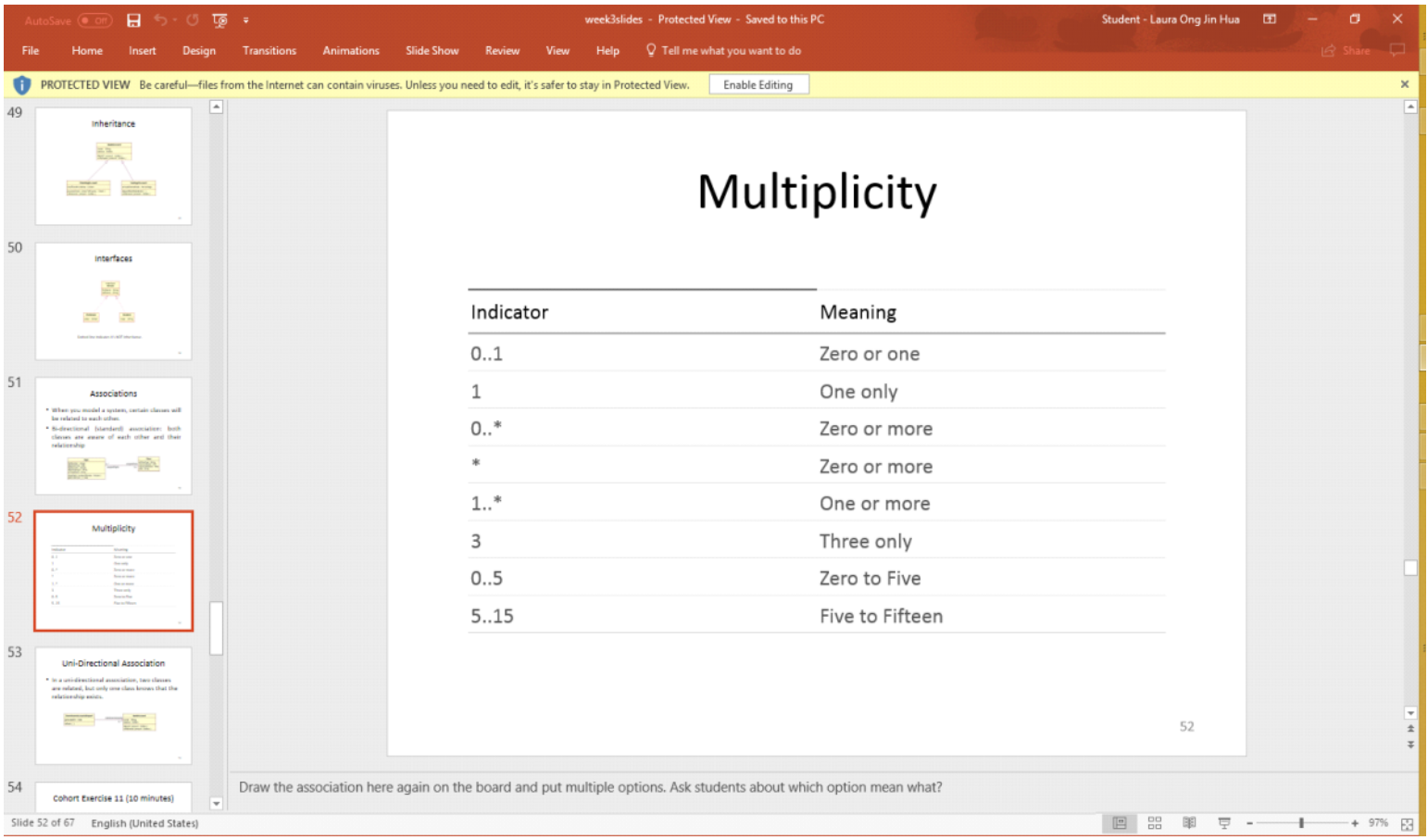
You see that these three are actually sending messages to subscriber. And these two are sending message to remote

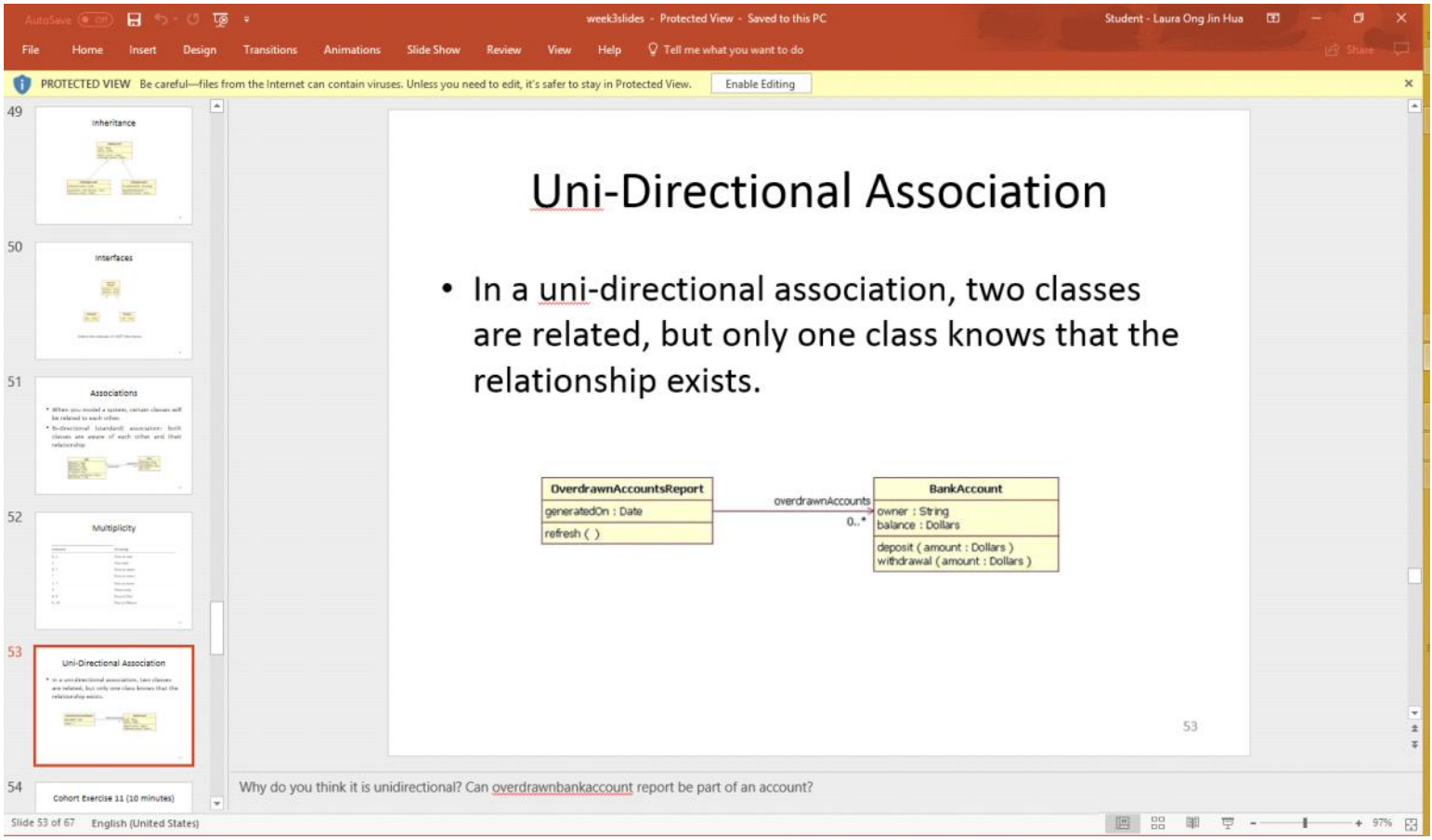


For every use case, I have a function in the class which is implementing the use case. I do not have any function in the clas s which is not in the use case.
Associations: When you model a system, certain classes will be related to each other.
Bi-Directional(standard) association: both classes are aware of each other and their relationship
I.E. plane and flight needs to know the existence of each other. Both classes are aware of each other. Flight knows which pla ne it has been assigned to. Plane knows which flight has to be known.
Then comes the attributes: number of the plane e.g.

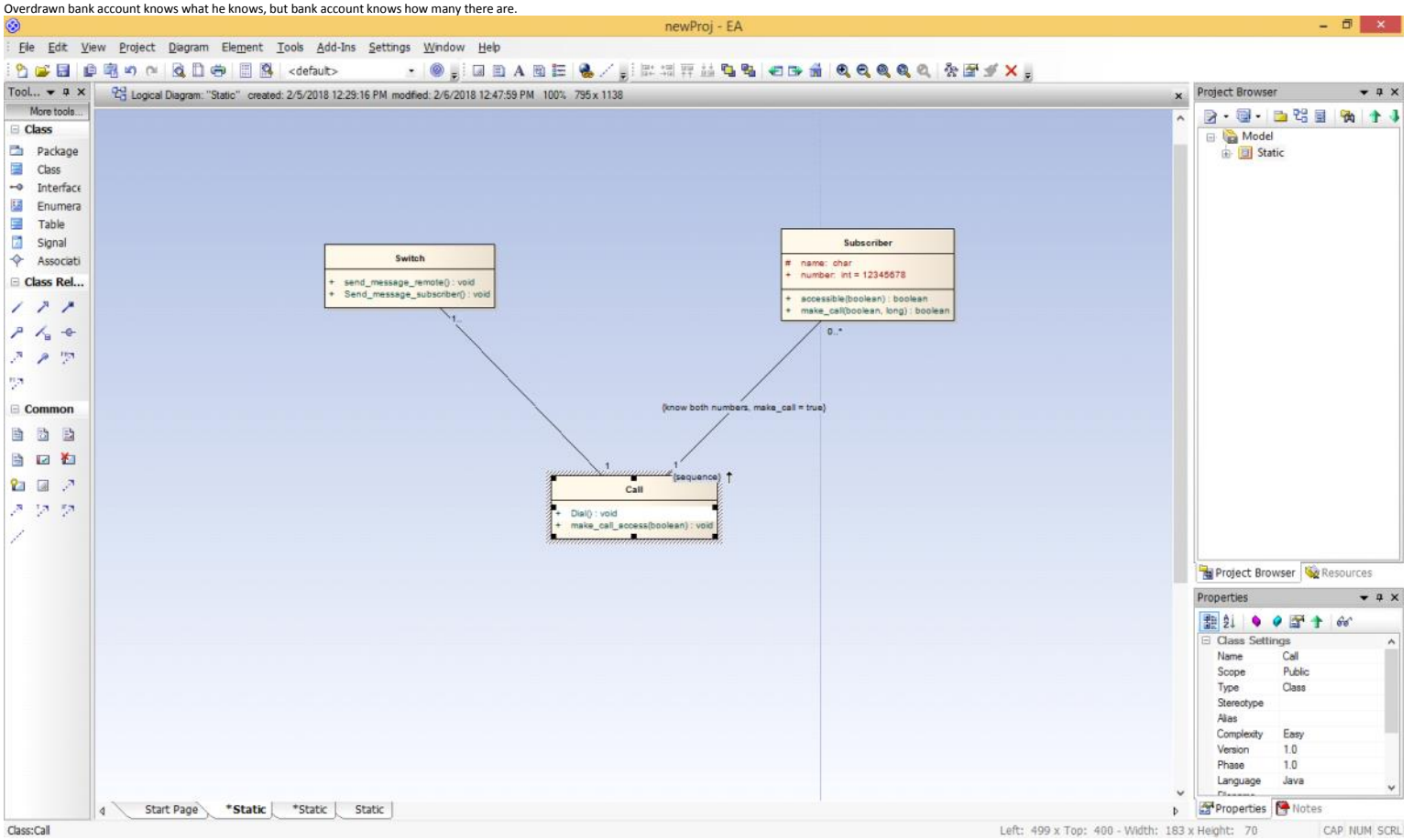
1. Whether uni-directional or bi-directional
2. What is the number of objects assigned to

I.E. It's more important to understand how many planes to oine flight path.
0..1 : Zero or 1
Constant: One only
If I assign a plane to three different flights. So you could have 3 on the plane end of the association line.
Association could be both uni and bi. Uni: two classes are related, but only one class knows that the relationship exist.



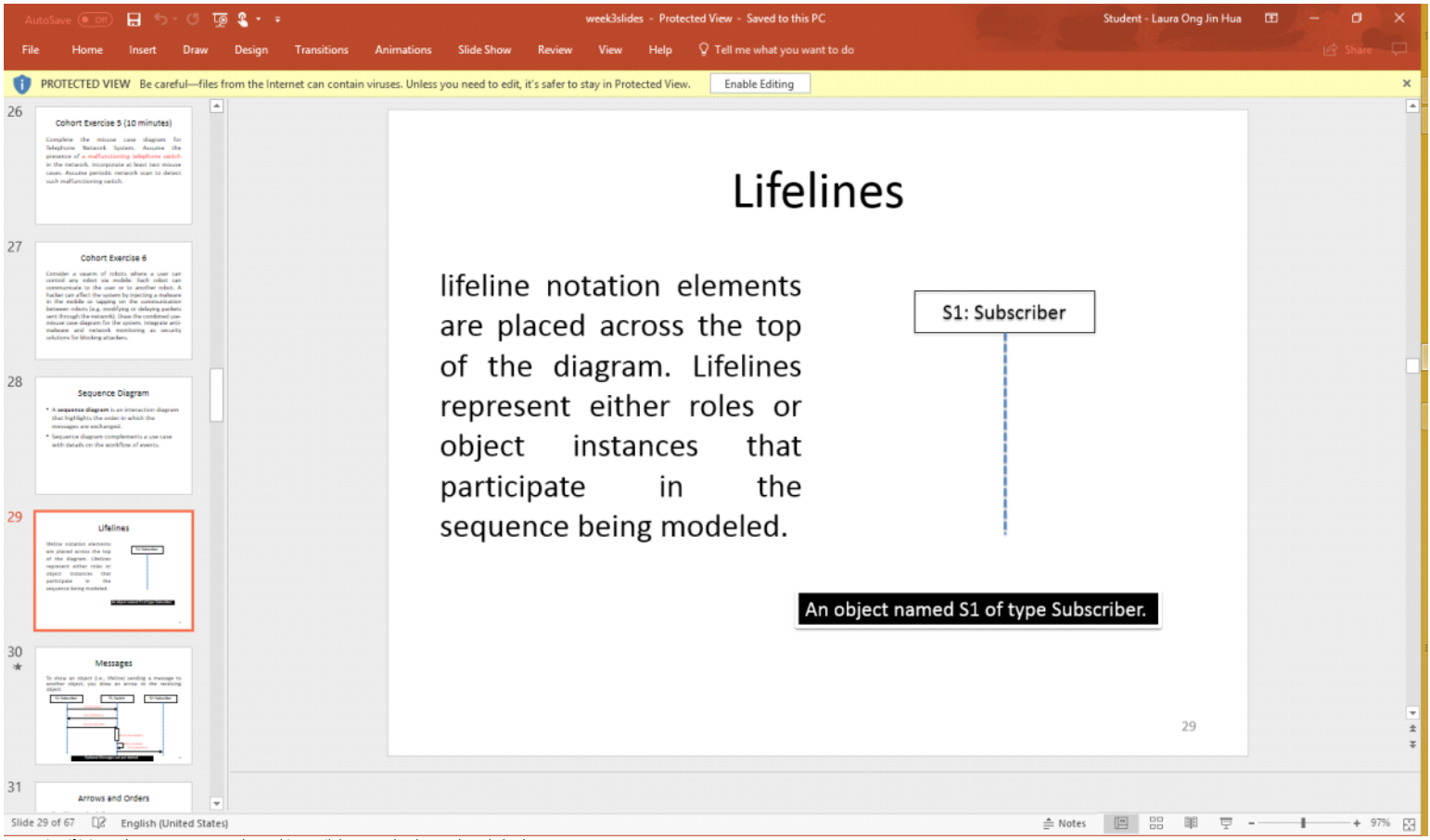


Overdrawn bank account knows what he knows, but bank account knows how many there are.

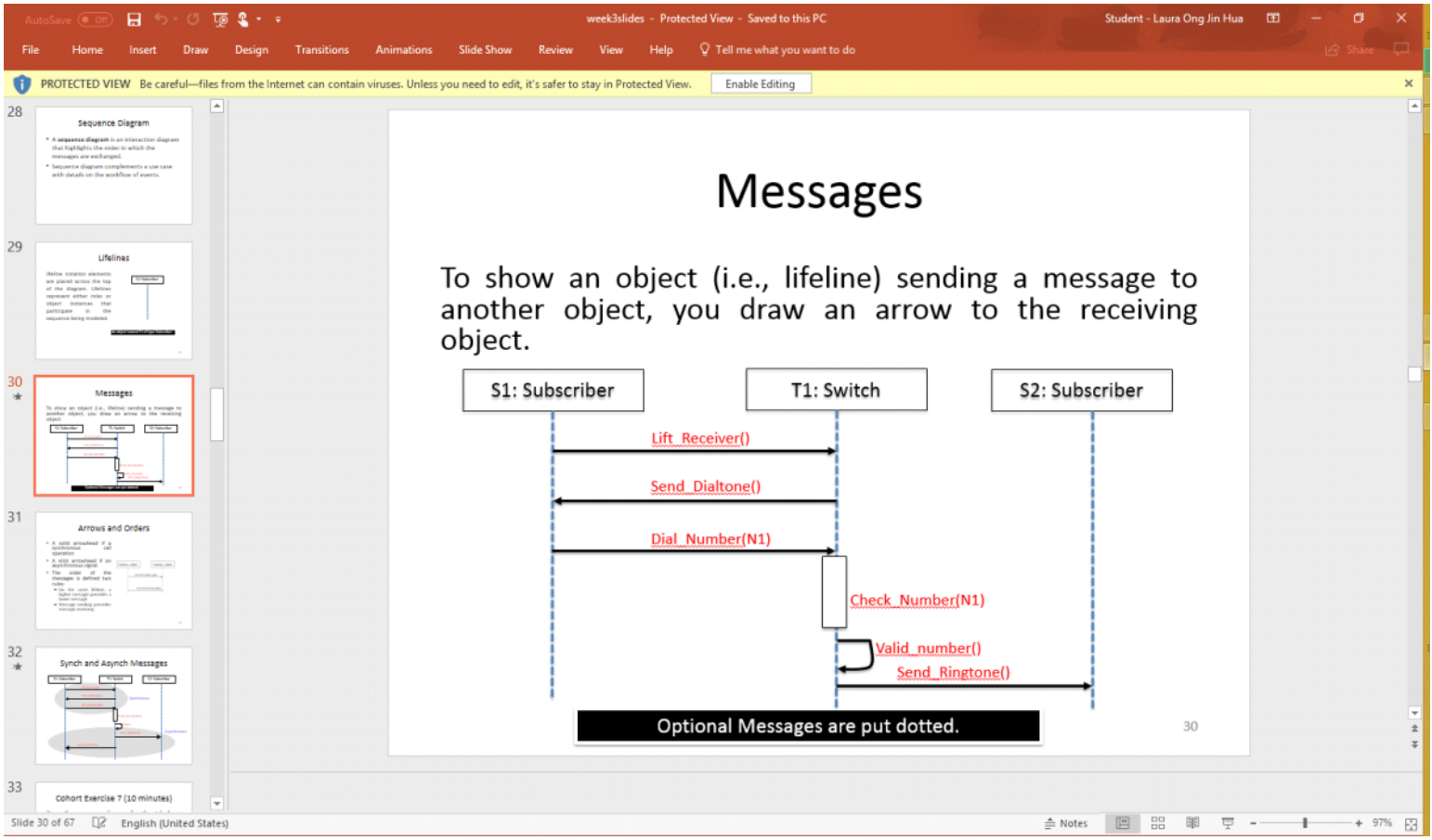


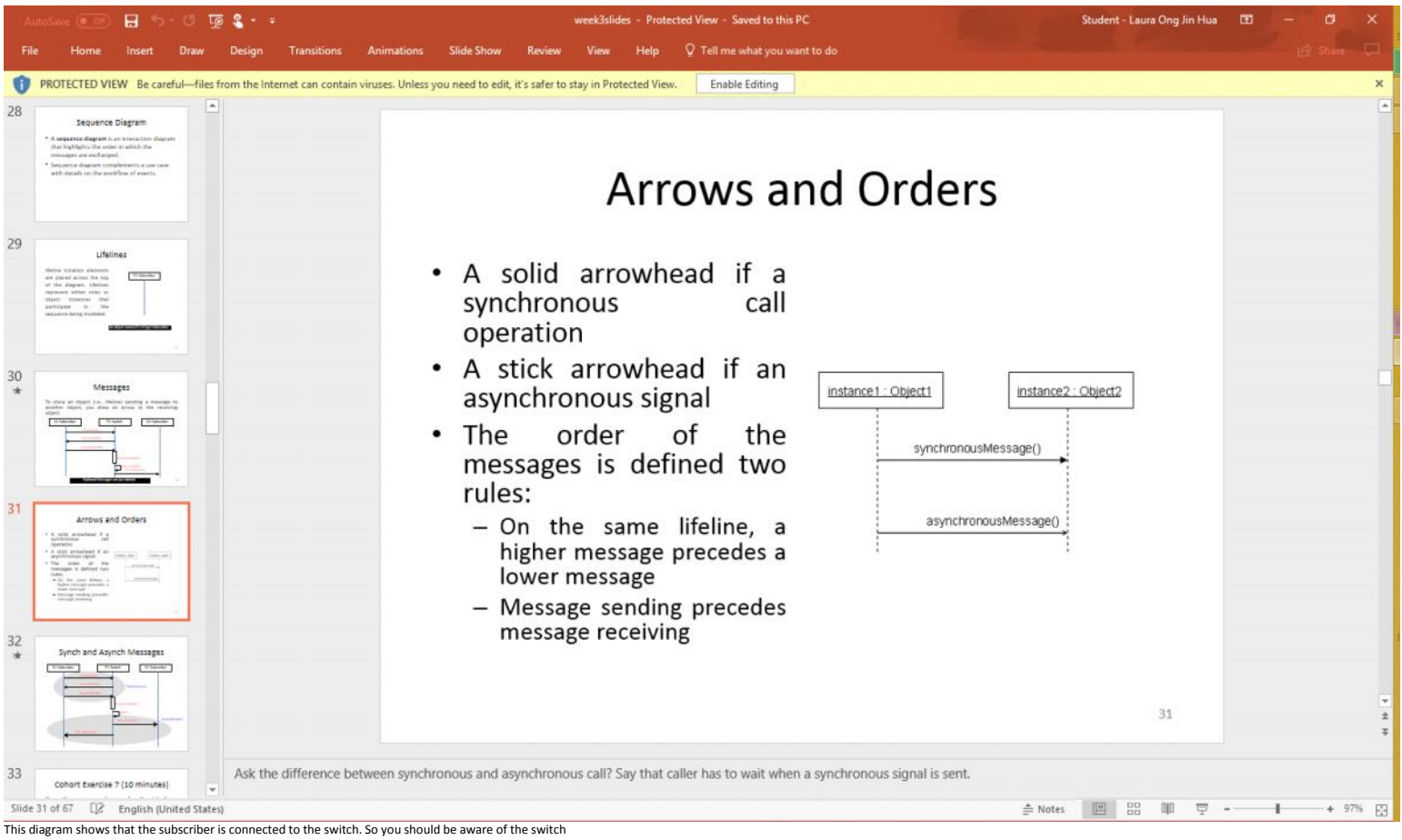
Friday

Friday, February 9, 2018 9:40 AM



In practice: if it is synchronous you cannot do anything until the return has been acknowledged.
However, for dial numbers, it is asynchronous and you can do stuff to it.





This diagram shows that the subscriber is connected to the switch. So you should be aware of the switch that you are connected to. If you have drawn the class diagram where it does not show any association between subscriber to switch, then you cannot have this kind of message. But actually, if you have bidirectional, then it is fine.

This association show that your subscriber can only talk to one switch at a time. One subscriber is

sending message to only one switch. It is not sending to multiple switches. If this subscriber was also sending messages to another switch, this diagram becomes inconsistent. You need to be careful about association. Subscriber can only send to at least one switch.

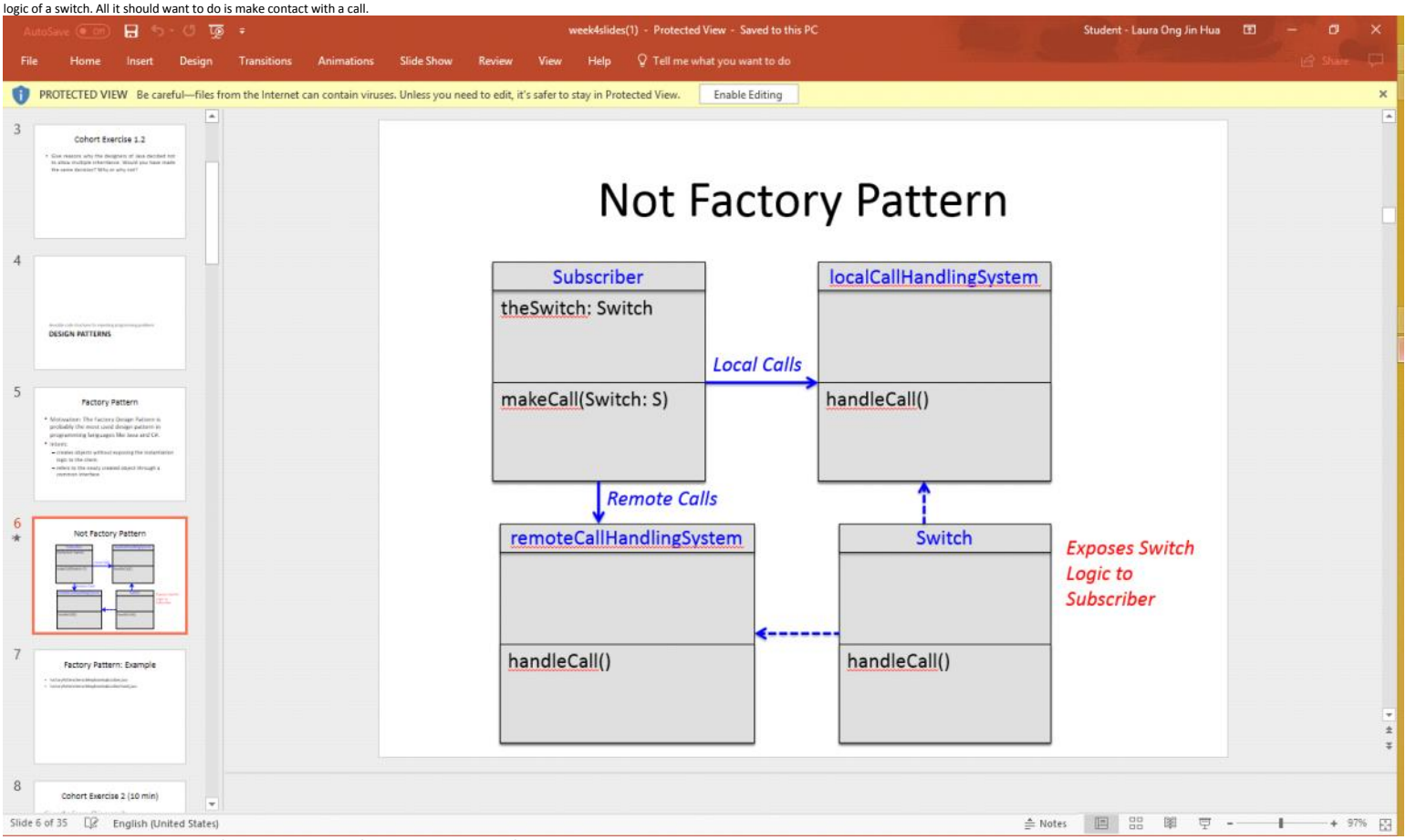
When you draw for big project, this is something to be careful about.

You draw state diagrams for each of your class. You drwa state diagrams for each of these classes.
In state diagrams you have states and transitions.
You have initial states and final states like this.

Monday

Monday, February 12, 2018 11:35 AM

Not a Factory Pattern: Why is this Switch not? The subscriber should not be aware of implementation logic of a switch. All it should want to do is make contact with a call.



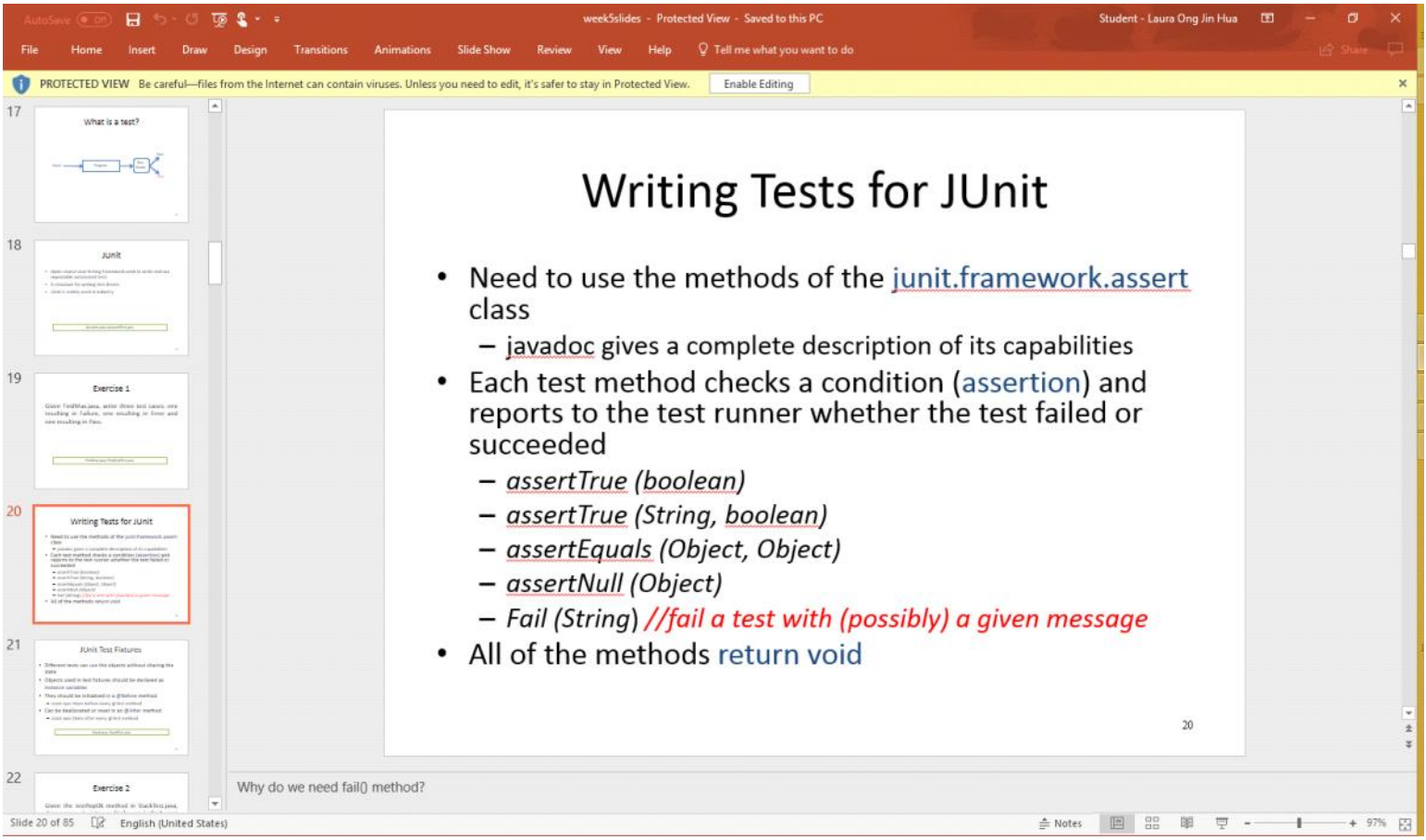
In Factory Pattern, I just create a switch factory, which will havea function called make switch for this call, for this call option (either remoteCall or local call). You don't instantiate the switch directly. Subscriber should have no idea of the different logic for local or remote. In its own logic, it will create the instantiation for the switch.

```
class SwitchFactory {

	// sudiptac: Encapsulation of all different subsystems in the switch
	public Switch makeSwitchForThisCall(String callOption){
		Switch newSwitch = null;

		if (callOption.equals("L")){
			return new localCallHandlingSystem();
		} else

		if (callOption.equals("R")){
			return new remoteCallHandlingSystem();
		} else
			return null;
	}
}
```

You have a switch factory (takes input from subscriber) This code was directly in the subscriber. Put it into a switch factory and subscriber only aware of factory class. Can handle both remote and local calls. Make the implementation transferent. There is an abstract class for the switch, and two different class which will be deriving from the switch class. One for local and one for remote.

Monday

Monday, February 19, 2018 11:55 AM



Fail : To show the developer was not implemented

Monday

Monday, February 26, 2018 11:41 AM

d >= e

|true	|false
Throws	SOPL
Error	e-d > f

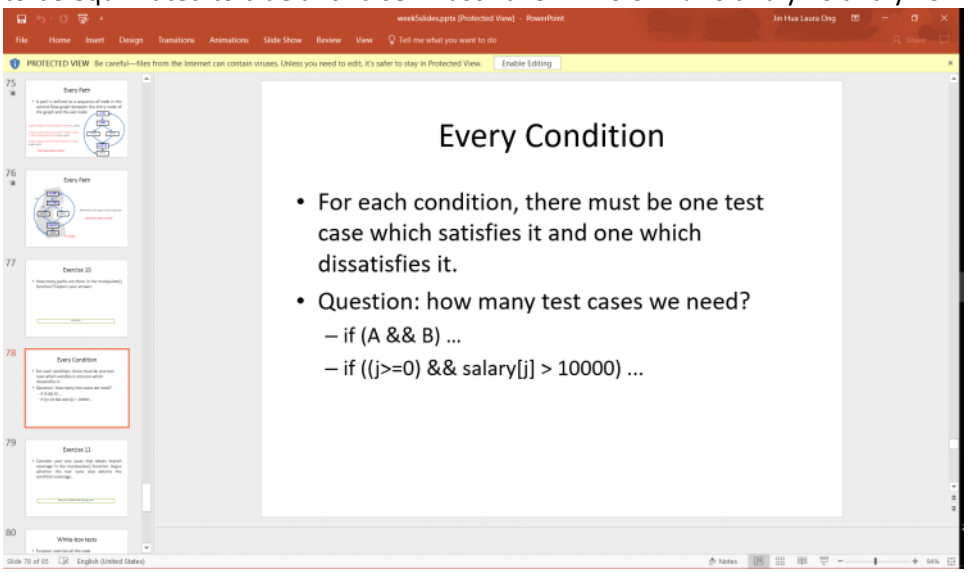
True	False
SOPL("once") Middle = (e+d)/2	SOPL("return") Return (e+d)/2
Middle < true	

True	False
System… D = middle	System… E = middle

Loop Tail back to e-d>f
This is called loop backage

Control Flow Graph: Captures how your program behaves at run time. After this block of code is executed, when you enter the control program for your code. Writing tests for coverage becomes much easier. So the point is that you must understand how the control graph is drawn from an arbitrary piece of code.

For each atomic condition; do not confuse with conditional statement. But where x == 0, for every test to be equivilated to true and false. Must have x == 0 or x != 0 and y <0 and y> 0.

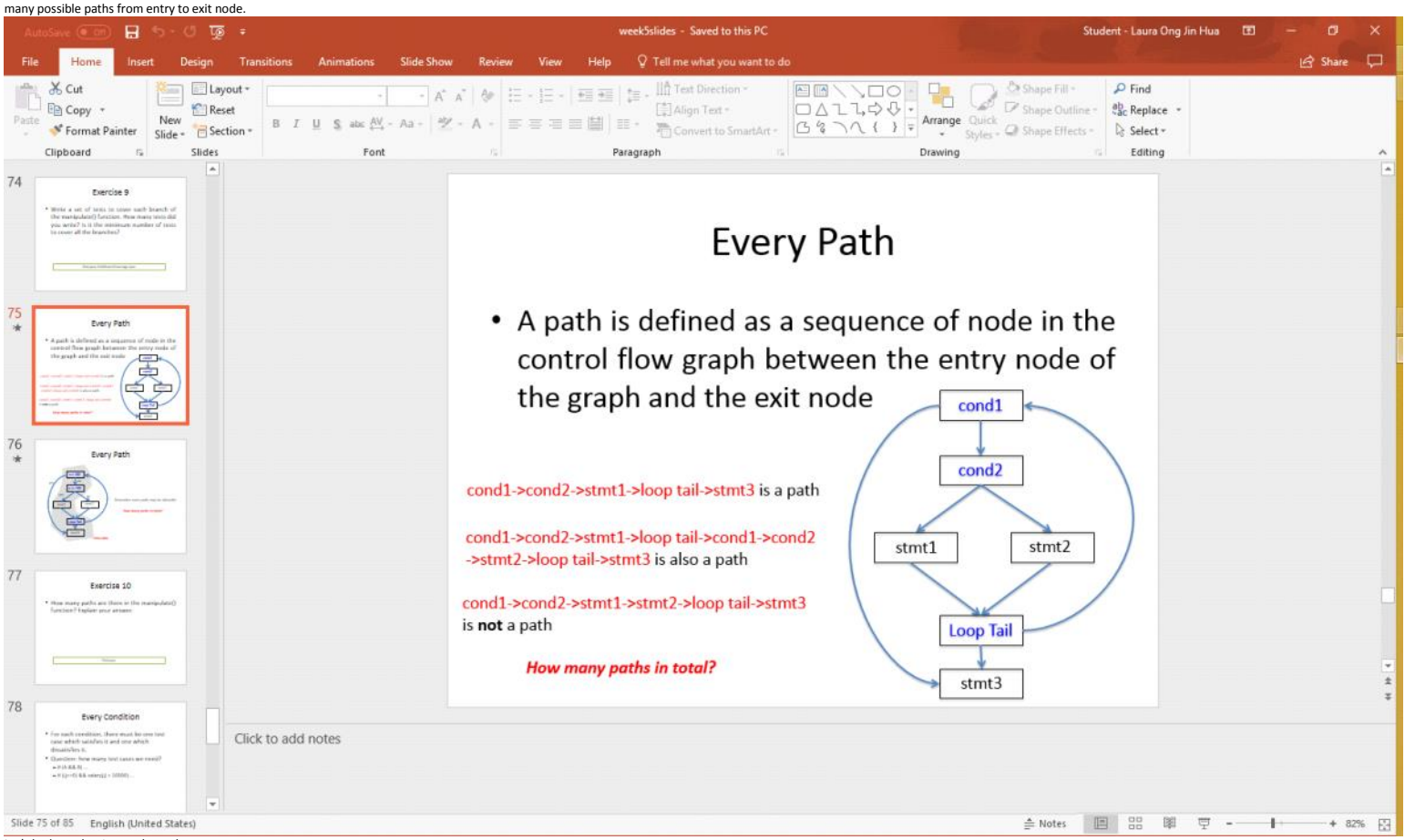


For a = t and b = f, a = f and b = t. You have condition coverage and no branch coverage. You only execute the false portion of the branch.

Tuesday

Tuesday, February 27, 2018 11:41 AM

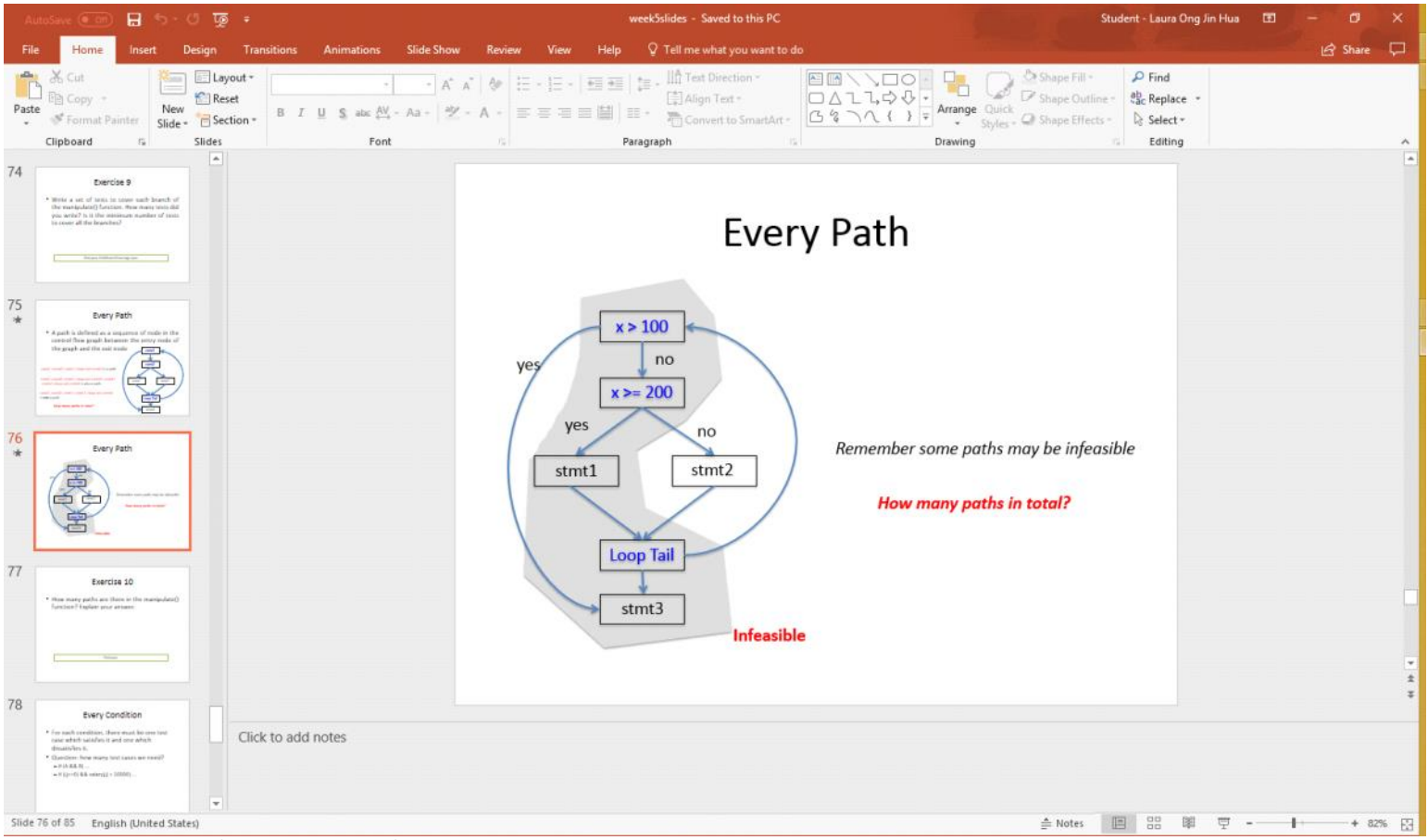
Most of the time you cannot achieve it : Path coverage.
A path is defined as a sequence of executed nodes in the control flow graph between the entry node of the graph and the exit node. It is a path which is representing the sequence of executed nodes in run time. Where the program actually enters and the exit node. So you can have multiple exit nodes. It is defined as a sequence of nodes between entry or exit. Since program has loops and branches, there are many possible paths from entry to exit node.



Let's look at what I mean by path.

There are two different paths. You execute the loop twice

Assume the loop is run through hundred times. You don't need to consider if one time or two times. So it is 2 ^ 100.



Not every path may be executed. Even for a realistic program you may find many paths that may not be executed.

Thursday

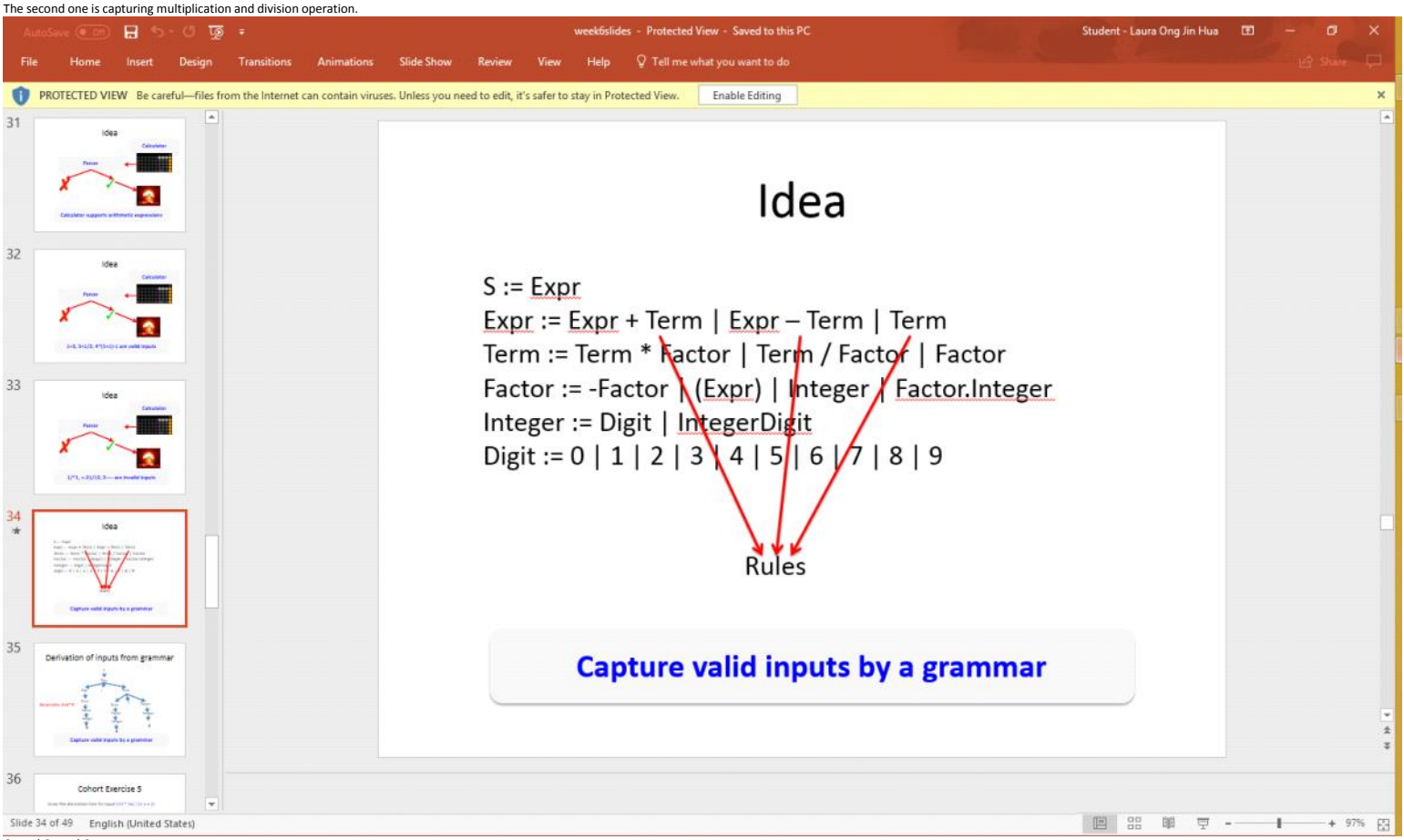
Thursday, March 1, 2018 10:41 AM

These are not valid input, but you might generate input that are invalid and there is no point in testing them because that is a parser and they are never going to reach calculator program. The idea is how we can mask this one and generate input to the calculator and go into the tick mark and go into the calculator to make it crash.

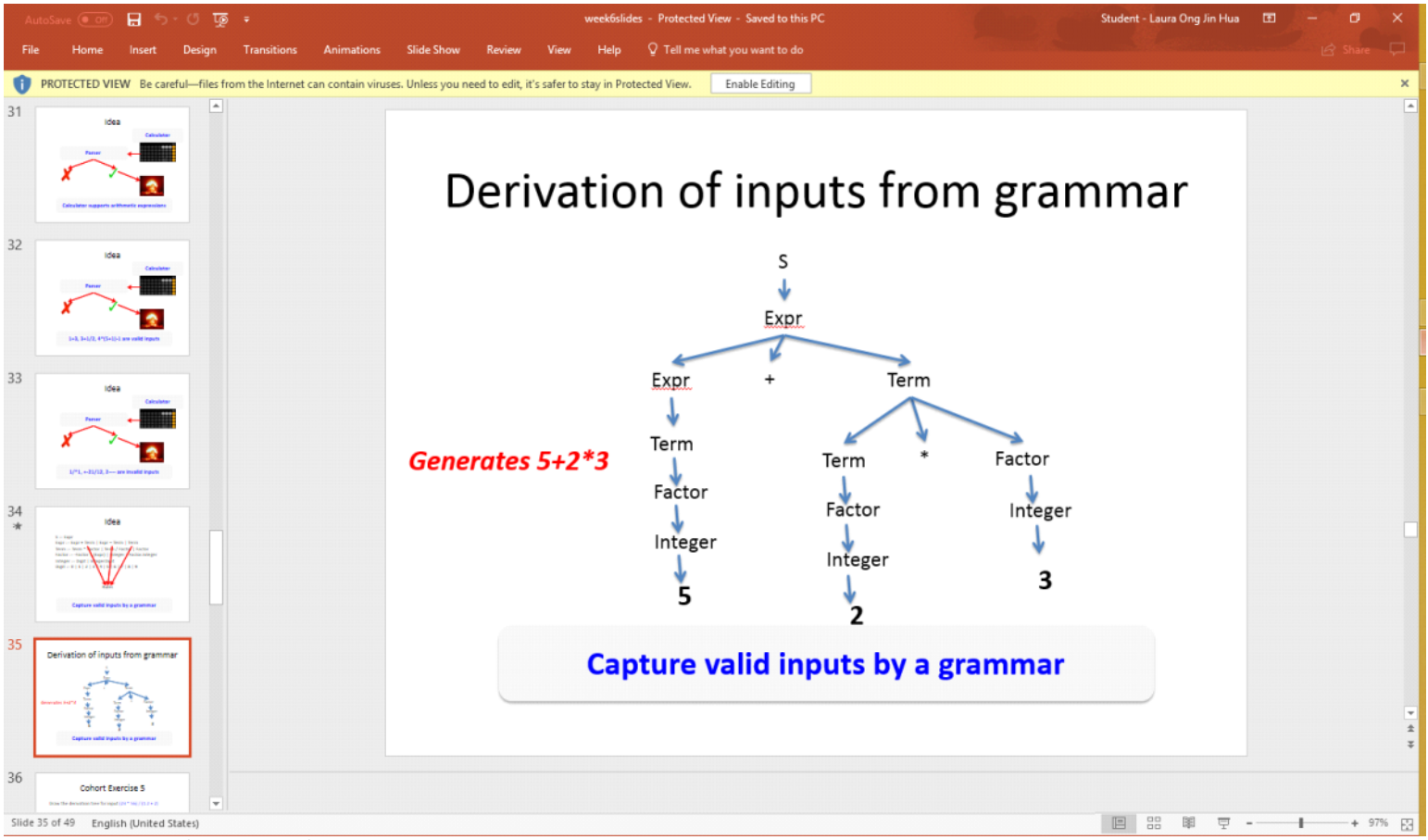
As long as you have the valid input for arbitrary program, it becomes easier to generate the valid input. All possible input are valid for the calculator. You can satisfy by grammar.

Grammar: can specify more languages like regex. Grammar : set of arithmetic expression is like the regular language. You can express any arithmetic expression. Whatever it generates it creates arithmetic expression. This part of the grammar is capturing addition and subtraction operation inside the grammar.

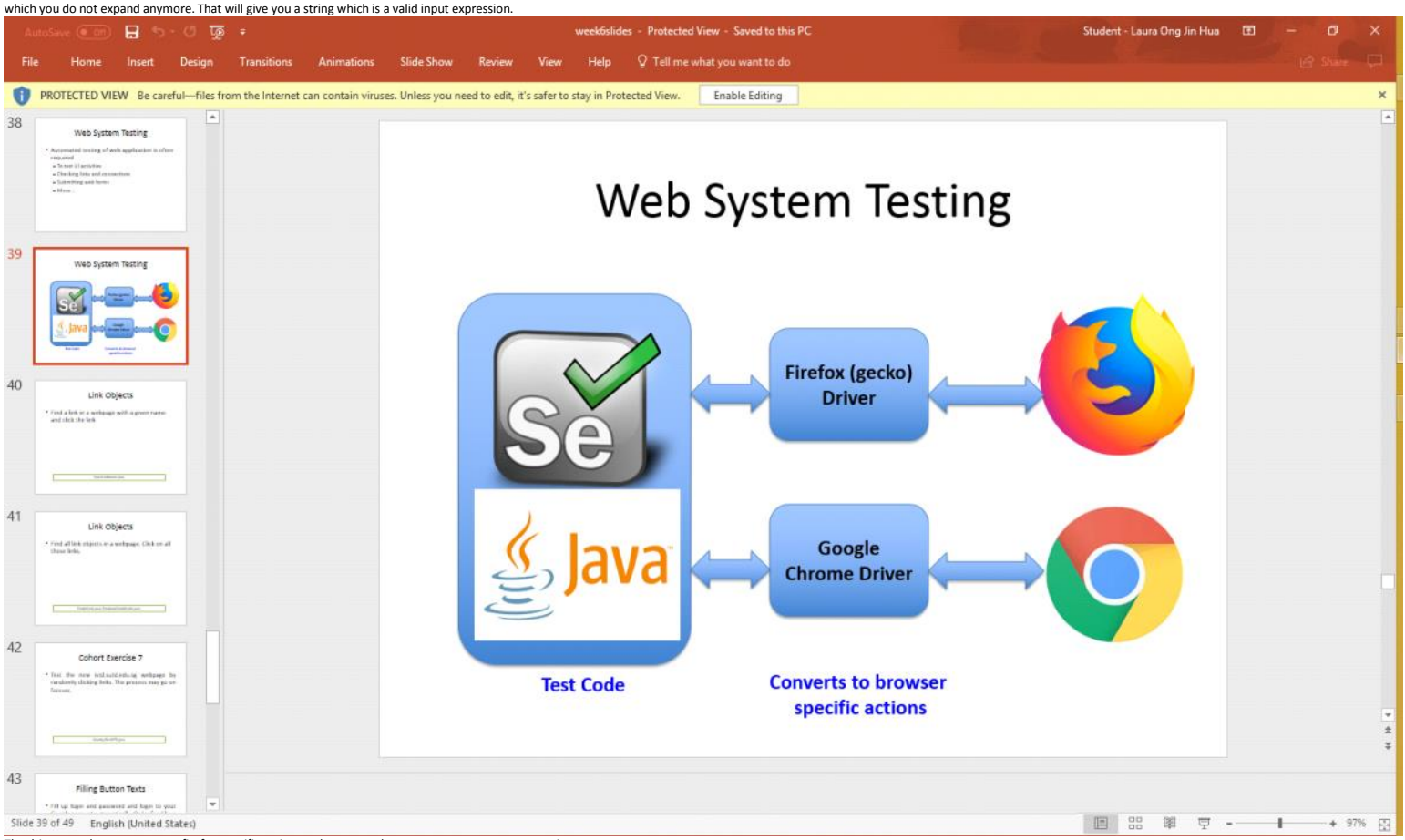
The second one is capturing multiplication and division operation.



Or and Or and Or.



Since I already have a valid input format; it is harder to find the bug. Given you have the grammar, how can you leverage the grammar. That's what you need to do for the next exercise. You need to random inputs for the calculator but you need to make sure all the input you generate are valid expression. Only when you have valid input format. You start with the symbol and start with some at random. After which you do not expand anymore. That will give you a string which is a valid input expression.



The driver translates program to firefox specific action, and execute whatever you wrote as a program in

firefox and done as a java program and actions as a driver.

What I mean by different element in a web page? You have text boxes and buttons.
How do you locate links in an arbitrary web object. We can write a program to do this for us.

All the things we learn about Junit and Jmock they are test drivers. They are not solving their work of test design. Test design is harder. These are tools for automating test design. There are three types of framework, how we can solve the framework. Selenium has many languages. With some modules not implemented. Junit is for unit testing. It's very important and much harder problem then using these two. When you have the executable problem and don't have source coe. We have two ideas: How you partition the input. Every test in one partition behaves differently. This requires some specification.

Jmock is useful because it does not require the class to be in place while testing. What it needs is an expectations() type returned whether it is true or false.

Monday

Monday, March 12, 2018 11:44 AM

Monday

Monday, March 19, 2018 11:51 AM