Categories/Things to note:
1. Timestamps (not needed yet)
2. Mental Models - what do they think (deductions)
3. Questions developers ask (includes hypotheses which should be labeled as such; hypotheses are defined as testable questions - need to go back through the questions and label some that I have missed)
4. Coding activities - what do they do to test ideas?
5. When do they reject hypotheses and why?
~~6. Goals and subgoals [merged with questions because they were too similar but left to keep numbering consistent]~~
7. Learning Activities
8. Miscellaneous (Other contributions to the solution algorithm)

Key/Guidelines:
-*Italics* means interviewer's interpretation without evidence (speculation)
-Hypotheses should be able to be answered with yes/no questions
-All statement should include who performed the action
-directives should be written out where they are discovered
-code up more and then decide if you should combine questions and goals (is there a goal that cannot be written as a question?)
- questions should have question marks at the end
- not capitalizing statements

Participant 1, Task 5, Fault 63

8) the participant reads through the task prompt and follows the instructions to cause the error.
2) while reading the prompt, the participant reads the statement about the error not occurring if the user doesn't navigate to HeadsUpFragment first and deduces that navigating to HeadsUpFragment causes the error
3) hypothesis - "So something about HeadsUpFragment causes the error?" - the participant doesn't immediately investigate this idea. The participant's reasoning (quoting email) - "I thought the error's stack trace would make it quick to accept or reject my hypothesis about it being something in HeadsUpNotificationFragment. When it didn't immediately confirm my suspicion, I realized I didn't fully understand the error message so I wanted to learn more about it. I always do this when I write code because I like to fill small gaps in my knowledge right away when I find them."
7) the participant reads through the error message
2) The participant sees the error message and decides to look into the Android lifecycle - here is his reasoning (quoting email) "The words "illegal state" in the exception message signaled that a lifecycle was in play to me when combined with the nature of

the bug.The words "illegal state" in the exception message signaled that a lifecycle was in play to me when combined with the nature of the bug.”
3) hypothesis - does the fragment have a lifecycle error?; specifically something to do with the active state
7) the participant reads in fragment documentation about fragment lifecycle - this fails to produce information on the active state
7) the participant searches in fragment documentation for keyword “active” - fails to turn up information on the active state
7) the participant returns back to reading the fragment documentation to find something about the active state in the fragment lifecycle
    7) the participant checks to make sure that the lifecycle causing the issues is the fragment lifecycle and not the activity lifecycle
5) after scanning document, the participant decides that the lifecycle error hypothesis is likely not correct and moves on to examine other possibilities
2) Participant stops looking through docs and begins to explore the code; here is his reasoning (quoting email) “Simply that I felt that my time with the docs gave me enough info to start looking at the code closer. If I had already known what I learned from looking at the docs I probably would have gone straight to the code.”
7) the participant looks into the code, specifically the HeadsUpFragment
    2) the participant determines parts of code would not have been executed because, during earlier program execution the participant only opened the tab corresponding to HeadsUpFragment
5) the participant doesn’t see anything out of place in the HeadsUpFragment initialization
7) the participant performs manual dataflow analysis based on error message - starting with looking into what happens when the participant clicks the change color button
2) the participant sees a line with the text green and the button name and deduces that line should be setting the button color - colorArgs.putInt(“ButtonColor”, Color.Green);
3) what line does the error message say caused the failure?
7) the participant checks the error message and finds that it was the next line, headsUpFragment.setArguments(colorArgs);
2) because HeadsUpFragment is referenced in the line, the participant decides to investigate HeadsUpFragement (already deduced HeadsUpFragment may be a problem earlier)
7) the participant tries to look up setArguments
    7) the participant starts by investigating HeadsUpFragment since that is the type of the variable
    2) the participant doesn’t find the setArguments method in HeadsUpFragment and realizes the method may be part of the Fragment superclass
    7) the participant pulls up the class structure to view class specific methods and doesn’t find setArguments
    7) the participant confirms that setArguments is the method he was looking for in HeadsUpFragment

7) the participant starts to look up the method in the fragment documentation

7) the participant checks the superclass of HeadsUpFragment is actually fragment and then returns the screen back to the failing line before starting to read the documentation again

7) the participant looks up setArguments in the documentation - and identifies the problem is directive **#63: setArguments can only be called before the fragment is attached to the activity** - but doesn't know where to go from there - is frustrated with the documentation because it doesn't propose an alternative

3) hypothesis - makes a wild guess; although it may be what he considers the least amount of change required for a solution - "Maybe I can get it's current arguments and change its current arguments?" (to create a solution)

7) the participant decides to look through the documentation to find if the participant can get and set the arguments; participant knows the participant has access to setArguments since the participant just looked it up so the participant is looking for getArguments.

8) the participant comments about difficulty with Android documentation structure; specifically looking for a method in the API - "Is there a way to get it to show me all the methods at once?"

3) hypothesis - "Maybe I can just use getArguments?" (to create a solution)

7) the participant reads the getArguments documentation

4) the participants determines the participant should get a Bundle returned so the participant will changed the returned Bundle and then set it again - the participant decides to modify the Bundle in place

4) the participant realises the test doesn't work so the participant restarts the emulator to see if that may be the problem (28.45).

7) the participant determines that the participant's change only removed the crash error but does not cause the button to change colors. The change also still kept the originally correct change color behavior: the color changes if the change color button is pressed before HeadsUpFragment is initialized. This means the participant found his change didn't break the application

2) the participant states that he fixed the crash but didn't fix the part that updates the button color.

7) the participant decides to look into where the change color button action occurs in change color fragment again

3) "I wonder if this get method is creating a new one. So I'm changing it one the new one but not actually on the one that already exists" - essentially is the getMethod not updating the arguments?

7) the participant decides he needs to understand the getHeadsUpNotificationFragment() method so the participant goes to the declaration and reads what happens to make sure the method is not recreating the fragment - finds that the method is not recreating the fragment

8) “I thought the documentation would have told me something about this not getting updated” - referring to the return of the getArguments method (32.01- not necessarily the beginning)

3) the participant wonders if the putInt method is not replacing the value

7) the participant decides to read the documentation on the putInt method to read what the method does

5) the participant reads in the documentation that the putInt method replaces any old value so the participant concludes the value is being replaced

7) the participant decides to look at another method call in the problematic line - getArguments()

7) the participant does not find the button color key defined in the Java methods in the project and concludes that it must be defined in the resources.

2) the participant remarks that he is unsure where the button color key comes from after looking through the Java files and the head up fragment layout file; also mentions that he isn’t sure that knowing where the key came from would help him figure out the problem

3) “would it cause it to be different if getActivity is different for some reason?” (meaning is a different return from getActivity causing the problem?)

7) the participant looks into the code and sees that getActivity is coming from the fragment class

3) “so fragments have different activities? That wouldn’t make sense would it?” (he asks himself as he continues to look into the getActivity call)

8) the participant spends a significant time looking into the API guide to find the method documentation when the participant was trying to look into the API reference - confused by the different documentation types

7) the participant reads through the getActivity method definition which doesn’t completely answer his question

3, 5) “this makes me feel like they must be associated.., why would they be associated with different activities” - expresses frustration that the fragments being associated with different activities is unlikely to be true

3) “what about the way that these are created are making it so that if I do that click first, it’s different” - the participant hypothesizes that the the problem is caused by a difference in the way the fragments are being initialized

7) the participant manually compares HeadsUpNotificationFragment and ChangeColorFragment to see if he notices a difference

7,5) the participant remarks how both call the same getArguments method; which further solidifies rejecting his earlier hypothesis

5) the participant rejects the hypothesis that the problem is caused by fragment initialization differences because the participant discovers the real problem

7) the participant realizes that the button is only updated with the view is created so it is not updated when the change color button is pressed when the view is already created.

7) participant decides he should call setBackground color

3) participant wonders if he can get the button to change colors from while in ChangeColorFragment
4) participant writes out a method to update the button color by moving the original code to update the button and then calling the method in both cases when the button is clicked.
4) tested the application and found that his solution works (42.30)

Participant 1 Task 3 Fault 17
8) the participant reads through the prompt
8) the participant comments that these task require knowledge of lifecycles inside of lifecycles and jokes that the participant has difficulty keeping track of it all.
7) the participant looks at the running application to find the missing options menu
7) the participant looks at the handout provided to see if the handout contains any clues because the handout had a question about the options menu
7) the participant decides to look for where the code is trying to create the options menu, since that was mentioned in the prompt - finds onCreateOptionsMenu in HeadsUpNotifiicationFragment
3) Hypothesis - does the menu.clear() call in onCreateOptionsMenu cause a problem?
7) the participant sees a menu.clear() call and the participant decides to investigate that call first
    7) the participant looks up the method in the documentation
    8) the participant expresses frustration that the documentation is not like JavaDocs (due to difficulty of finding a specific method call)
    5) the participant reads the method documentation call and decides that the menu.clear() call should not be doing anything problematic
7) the participant goes back to looking at the onCreateOptionsMenu method
7) the participant determines that onCreateOptionsMenu is missing the override statement after discovering that the Fragment class also defines onCreateOptionsMenu
    3) hypothesis - not having the override call prevents the method from being called or the error being displayed
    4) the participant adds the override statement to the method
    5) after adding the statement no compiler errors occur
    2) the participant remarks that he really expected to get an error after adding the override statement
7) the participant looks up the onCreateOptionsMenu call in the documentation
    7) the participant confuses the onCreateOptionsMenu call for an Activity in the documentation with the onCreateOptionsMenu call for a Fragment in the code and spends time figuring out why the methods don't match
    7) the participant discovers the directive, **you must first call setHasOptionsMenu(true) before using onCreateOptionsMenu**, in the documentation for the onCreateOptionsMenu method
3) Hypothesis - "setHasOptionsMenu" is not being called first

7) the participant searches through the project for the method call using find; find returns that setHasOptionsMenu is not in the project
2) the participant concludes that the lack of setHasOptionsMenu in the project is a problem
7) the participant looks up setHasOptionsMenu in the documentation
2) the participant remarks that he is unsure where the add the method call in HeadsUpNotificiationFragment
3) the participant states a guess about adding the method in onCreate but mentions it may not work due to lifecycle dependencies - hypothesis: “the setHasOptionsMenu call should be added in onCreate in HeadsUpNotificiationFragment”; here is the participant’s reason for adding the method in onCreate (quoting email) “setHasOptionsMenu was the only way I knew to tell the framework that the activity had an options menu and it seemed like I wanted to do that as early as possible in the lifecycle. From my reading of the docs earlier in the video I knew onCreate was the earliest possible opportunity.”
4) the participant adds the method call and then tests the application
7) the participant waits a long time for the application to refresh because the participant forgot to select the device to test the application on
8) the participant remarks that it takes a while to test the application
4) the participant finds that his solution works

8) between tasks the participant remarks that the previous task had less parts to achieve a solution than the first task

Participant 1 Task 7 Fault 12
8) the participant reads through the task prompt
7) the participant executes the steps in the application to cause the application to crash
7) the participant runs the test cases to see the test case error
8) the participant remarks that AndroidStudio has a more subtle way of notifying developers if all tests passed than Eclipse, which makes running test cases in AndroidStudio more difficult.
7) the participant didn’t find the IllegealStateException message from the test cases useful so the participant decided to run the application again to see the error stack trace (1.06.21)
7) the participant decides to investigate OtherMetadataFragment since that fragment caused the crash
7) the participant determines that the last line the participant was executed successfully was in the onCreateView method since a debugging statement was left in the method
8) the participant comments that the exception does not seem to have a useful stack trace at all, since it is all Android internal method calls
2) the participant decides that OtherMetadataFragment may be failing due to something it is doing differently than the other fragments

3) hypothesis - Is OtherMetadataFragment failing because of a unique implementation that should be consistent across all fragments in the project?
8) the participant starts by comparing the failing method, onCreateView, in OtherMetadataFragment with the onCreateView method in VisibilityMetadataFragment (1.09.10)
2) the participant notices that inflator.inflate is different between the two methods; participant sees the directive issue, **if using inflate() in onCreateView, then pass False to inflate()'s third argument**, and comments on the directive, but doesn't understand the problem.
7) the participant decides to look up the LayoutInflator inflate method in the documentation (1.09.50)
7) the participant becomes confused with the LayoutInflator.filter interface in the documentation for a bit but realizes the participant needs another class's documentation
7) the participant reads the LayoutInflator class's inflate methods documentation and comments that they are confusing
4) the participant decides to add the false parameter to the inflate method in the OtherMetadaFragment
4) the participant tests the change with the test case and finds that the application now works
4) the participant tests the change in the emulator and finds that the application works
2) the participant comments that he is unsure why adding the false parameter to the inflate method fixed the issue. The participant comments that the participant only made the change because the files were different, and a common debugging strategy is to change something that isn't working to match something that is working.. The participant comments that the documentation doesn't explain why making this change fixes the problem. The participant also comments again that the error stack trace was unhelpful.

Participant 1 Task 2 Fault 29
8) participant reads the prompt
7) participant checks out the working application to see the times in the executing application
8) at around 1:18:13, participant asks (quoting email) "It doesn't do this for me?"; quote from participant about what the participant was referring to "It seemed strange to me that I was using a specialized IDE for this specific framework and it would let me make a coding error when it could automatically add a default constructor for me. Like, the constructor is obviously a requirement and the IDE is obviously telling me exactly what the problem and solution is, why not just add the required constructor with a todo comment or something to fill on the body immediately when the error is known? I feel like I'd see the code change as soon as the constructor was inserted since it like"

7) participant reads serious warning about the directive: **This fragment should provide a default constructor (a public constructor with no arguments) (com.example.android.lnotifications.HeadsUpsNotificationFragmment)**

From the Fragment documentation:
Every fragment must have an empty constructor, so it can be instantiated when restoring it's activity's state. It is strongly recommended that subclasses do not have other constructors with parameters, since these constructors will not be called when the fragment is re-instantiated; instead, arguments can be supplied by the caller with setArguments and later retrieved by the Fragment with getArguments().
3) hypothesis - Does fixing the error as stated in the error message cause the fragment to compile and execute successfully without the serious warning?
2) the participant comments that error message is telling the participant exactly what needs to be done
2) the participant seems to get a better understanding of the error but is unsure how to implement the correct change based on that message alone.
4) the participant decides to add a Bundle object to the parameter list and deletes the Gregorian calendar parameter
4) the participant starts to add a getArguments call in newInstance method of HeadsUpNotificationFragment
    2) realizes that the method is static and decides to move the method call somewhere else
    4) participant changes the newInstance method to take a Bundle as the parameter
5) hypothesis - The GregorianCalendar instance must be passed to the fragment using a Bundle
    7) the participant looks through the documentation to figure out how to put a GregorianCalendar instance in the Bundle object
    5) the participant determines after reading through the class's method list that none of the Bundle's methods allow him to put an Object in the Bundle
2) the participant is unsure what to do be decides to try giving all the fragments the same Bundle
    4) the participant changes the code to use Bundle in the newInstance method for all Fragments
4) the participant tries to get rid of the error by creating a default constructor in OtherMetadataFragment (the participant changed Fragments - here is the participants reasoning by email "This is the only one I am having trouble reconstructing after this much time. I think based on standard OOP design principles I realized I was looking at the wrong class for the problem I wanted to solve. If that wasn't the case, it was simply that I was looking at HeadsUp while thinking but didn't really care about it, I was just looking at something compulsively like you'd tap a marker on a whiteboard while trying to solve a problem on the board."; that still didn't sound right based in context, because

both classes contained the exact same problem so I asked again and explained the context, here is the response “It makes me believe it's the OOP fork more, but I honestly don't recall this particular point in the exercise.” *So Zack doesn’t think that was the real reason. Zack thinks the participant had made changes to HeadsUpNotificationFragment that would be tough to undo, so the participant started on a fresh class*

7) the participant determines that this change fixes one of the errors in the file but doesn’t solve both

7) the participant has difficulty expanding the AndriodStudio error message

2) the participant comments that the participant read the error message wrong before

4) the participant removes the parameter from the newInstance method in OtherMetadataFragment

4) the participant comments out the non-default constructor in OtherMetadataFragment

4) the participant reverts LNotificationActivity to it’s original state

4) the participant then removes the arguments from the newInstance calls

4) the participant gets the arguments from the OtherMetadataFragment to set the Bundle in LNotificationAcitivity

4) the participant adds the time as a String to the Bundle

    3) “how do I get the time as a String” before quickly figuring it out

4) the participant changes OtherMetdataFragment so the fragment saves the time Bundle argument and later prints out the time.

7) the participant determines that the file does not have any more errors

4) the participant checks that the Fragment still works by running the application

    4) the participant waits a while because the participant forgot to decide which device to use

    7) the participant determines that the application fails with a null pointer

7) the participant reads through the error message generated by the failing application

7) the participant rereads the original error message in HeadsUpNotidiciatonFragment and determines that the participant should have added code to call setArguments(Bundle).

4) the participant changes the way the code was setting the Fragment arguments to use setArguments

4) the participant tests the changes and determines that the application is now working correctly.

Participant 6 Task 2 Fault 29

8) the participant reads through instructions

7) the participant runs the application to make sure it works and see the correct result - having the start date printed at the bottom of the fragments

7) participant reads serious warning about the directive: **This fragment should provide a default constructor (a public constructor with no arguments) (com.example.android.lnotifications.HeadsUpsNotificationFragmment)**

From the Fragment documentation:
Every fragment must have an empty constructor, so it can be instantiated when storing it's activity's state. It is strongly recommended that subclasses do not have other constructors with parameters, since these constructors will not be called when the fragment is re-instantiated; instead, arguments can be supplied by the caller with setArguments and later retrieved by the Fragment with getArguments().
3) hypothesis: does the application contain a correct implementation of passing an argument to a fragment?

5) the participant searches through the code base but does not find a correct implementation anywhere

7) the participant tries to figure out where the GregorianCalendar object is coming from to understand the how to pass the time correctly

7) the participant reviews the instructions in the error message

7) the participant reads through how the GregorianCalendar is current passed from the Activity to the Fragments

2) the participant comments that the participant is not Googling for help at the moment because the the participant is unsure about the keywords to use in the search

7) participant notices the Bundle parameter in the onCreate method

2) the participant questions whether that instance should be used in the constructor before deciding against it (25:50)

7) participant Googles for "android studio avoid non-default constructor in fragments" and selects a blog post link

7) the participant find the newInstance method and determines that the method already takes the GregorianCalendar instance.when the fragment is created (27:21)

4) the participant removes the parameter from the constructor and tries to figure out how to set the mAppStartTime variable in the newInstance method which the participant does by changing the variable type to static

4) the participant tests out the application and finds that the application is working

Participant 6 Task 1 Fault 31
8) the participant reads through the instructions and the examiner steps the participant through the error in the application to cause the error: "meeting set from default value to default value".

3) the participant wonders if the meeting information is being written anywhere or if the time goes directly to the screen.
3) the participant wonders if the problem is a simple example of the time data not being passed correctly
7) the participant clicks through the application to determine the fragment which has the error (the meeting fragment)
7) the participant begins to read through the meeting fragment
7) the participant rechecks what time selection fragment is doing
7) the participants checks setting the times in different orders to make sure the result is the same error
3) what is ft? - the participant is trying to understand the type of a variable
3) the participant questions what a FragmentTransaction is but decides to try to infer it from context
2) the participant doesn't see any variables saved to the global state in the for loop in onViewCreated and doesn't check the method calls, so the participant assumes that the for loop code result is not being saved
2) the participant realizes the method is building the buttons for the view
3) the participant questions why the buttons were built this way
2) the participant realizes that the participant needs to get the button values chosen to the print out location
7) the participant tries to understand how to get the data from the buttons
7) the participant decides to googles to understand the FragmentTransaction class
    7) participant reads the class description in the Android documentation and makes a sarcastic comment about the description being useful
7) the participant goes back to trying to understand the button initialization code to understand how to get the time data from the buttons
2) the participant deduces that the Bundle is a collection of arguments
**participant comments that he is still unsure how to get the time data from the buttons"
7) the participant looks up the commit message of the FragmentTransaction class and comments that the quick method summary is not useful and then reads the longer method details (41:06)
7) the participant tries to google TimeSelectionFragment to find an example
7) the participant finds the class in the project
2) the participant deduces that if the participant can get access to the TimePickerFragment, then the participant has access to the user selected hour and minute
7) the participant reads to understand how the TimeSelectionFragment and the TimePickerFragment interact
3) what does getFragmentManager do?
    7) the participant Googles getFragmentManager() in the documentation but then quickly changes to a StackOverfow question about an undefined FragmentManager. The participant decides the StackOverflow question is not

what the participant was looking for so the participant goes back to the Fragment documentation.

3) how are all the fragments in the code connected together?

7) the participant Googles Android Bundle and reads a StackOverflow answer on the question

7) the participant Googles FragmentManager beginTransaction and reads the method description in the documentation

4) the participant types TimePickerFragment tpf = getFragmentManager(). to check out the autocomplete options

7) the participant sees findViewById and looks up the method in the Android documentation

** the participant reiterates that the participant is trying to get a reference to the TimePickerFragments to get the time data from them - the participant does this multiple times throughout the experiment - not sure if it is important or if the participant is doing it for the examiner's benefit**

2) the participant guesses that the add and commit of the FragmentTransaction adds the buttons to a global pool where they can then be accessed later

7) the participant find an example of findViewById and assumes it will

7) the participant looks up the meeting button in the xml

7) the participant tries to look at the time selection fragment xml but gets confused by the fact that AndroidStudio displays a sample of the current layout instead of taking the participant directly to the xml

4) the participant completes the line the participant was typing earlier with findViewById() but then doesn't know the ID to find

3) how does the button model work in Android?

7) the participant reviews the TimePIckerFragment and TimeSelectionFragment to understand how they work together and fit in the MeetingFragment

7) the participant reviews the running application while looking through the code

2) the participant deduces that the TimeSelectionFragament creates a listener for the TimePickerFragment

4) the participant notices that the code contains a show call from the FragmentTransaction without a commit. The participant starts to add a commit call but stops when commit does not appear in the autocomplete options.

3) does the TimeSelectionFragment have the user selected time at the end of onViewCreated()?

3) what does the tpf.show call do?

2) the participant decides that the participant can pass the data from the TimePickerFragment to the TimeSelectionFragment

4) the participant decides to try to type tpf and look through the autocomplete options

3) what is the model for how Android passes stuff around?

3) is there a unique id for each TimeSelectionFragment? (!! - related to directive - !!)

2) **the participant notices there does not seem to be a unique id associated with each fragment**

7) the participant looks for a unique id for each fragment

7) the participant tests the application again to ensure the application is storing the times (1:00:16)

3) where is the button text set?

7) participant reads through to find the button set text

8) participant has trouble searching for where the setButtonText method is used because the project doesn't currently compile

4) the participant tries to use autocomplete to guess an ID for the findFragmentById call (the non-compiling line)

5) the participant deletes the line the participant added and decided it was not the correct approach

7) the participant goes back to looking for where the button text is set

2) the participant comments that the participant has a vague idea of the model but doesn't understand enough to do what the participant needs to do - "I want to understand the Fragment action model at a higher level" also (after the button initialization) "once this initialization is done, what is in the global scope, what can I pull things out of, what is a singleton sitting around somewhere vs. what is registering all these fragments and how do we access them", the participant summarizes his understanding of the application. The participant also comments that two TimeSelectionFragments are created but the participant is unsure how they are uniquely identified, so the participant can get the picked user data.

3) can the TimeSelectionFragments be found from a global registry?

4) the participant types getFragmentManager(). to see the autocomplete options

2) the participant comments that if the participant was really working on this project, the participant would take time to understand the event model, read a couple of tutorials, and look at a couple of examples

2) the participant thinks it would be useful to figure out this task since other tasks will probably use a similar framework'

4) the participant tries to complete the line started above with findFragmentByTag

7) the participant looks up the findFragmentByTag method in the Android documentation and finds that it is possible to add a tag when the fragments are added

3) "Maybe I need to tag these (the TimeSelectionFragments) whenever I am adding" them to the MeetingFragment

7) the participant looks up FragmentTransaction in the documentation to find an add method with a tag - finds the method

4) separates out the add calls and adds a tag for each fragment

4) tries to use findFragmentById to find the fragments in the scope to print out the results.

4)participant fixes a type error from the findFragmentById call

3) "do I need to cast this or does this not do what I think it does?"

4) the participant adds a cast to the calls (1:11:38)
4) the participant stores the hour and minute from the TimePickerFragment in the onViewCreated method
4) the participant adds a way to get the information from the TimeSelectionFagment in the MeetingFragment
4) the participant comments that the participant always feels like the participant has no idea what is going on when the participant starts with a new API
4) the participant has a compile error so the participant undoes the changes to the code until the file goes back to the way it started.
4) the participant tries to get each button using the unique tag that the participant re-added
4) the participant has problems referencing the variables in the button class.
2) the participant reads the error message that the participant needs to declare the variables final, but determines that the problem is that the time variables are private and not public, which is not correct and don't cause the file to compile
7) the participant rereads the error message and realizes that the participant adjusted the wrong variables
4) the participant declares the buttons as final
7) the participant runs the application and finds that the application crashes with a null pointer exception
2 the participant figures out that the application is crashing because of a null reference from the findByViewId calls
7) the participant goes back to reading the FragmentTransaction documentation but quickly goes back and forth between the two
2) the participant comments that the participant is confused about the Bundle term and what is retained
2) the participant realizes that the participant was getting the button references outside the listener
4) the participant changed the code to work in the listener and then the application works but the application prints 0's instead of the time value
3) is the problem something with the timeSelectionFragment or the picker not having the data at this point?
    4) the participant prints out the values of getSelectedHour and getSelected Minute when the button is clicked but before the user has chosen the time, although the participant thinks this location is after the user has chosen the time
    7) the participant executes the application and finds that the values printed are 0
3) hypothesis: is the problem that the there is a missing commit statement?
    4) the participant changes the commit call from the TimePickerFragment to the FragmentTransaction object
    5) the participant ran the application and finds that it throws an error that commit is already called. (1:33:22)
2) the participant considers tagging the different TimePickerFragments and accessing them directly.

4) the participant creates a setMinute and set hour method in the TimeSelectionFragment
4) the participant sets the TimeSelectionFragment minute and hour on time set
7) the participant tests the application to see what hour is being set to and finds that the hour is still set to 0
3) can the participant get the button text to get the set time? “How can i get the button text?”

4) the participant creates getButtonText method
4) the participant sets the button text to the different button text values
7) the participant tests the application and finds that it works

2) the participant comments that the participant spent a lot of time trying to figure out the timing of the parts of the application, how the fragments worked together, and how to register the fragments

(**Note participant 8 wants to see a copy of the final paper**)
Participant 8, Task 7, Fault 12
8) Participant reads through the assignment
7) “Let me see the error message first”, the participant starts investigating the fault by looking into the error message after the crash.
7) “So everything I just try and Google the exception, so that gives a good insight into it.” The participant Googles the stack trace line, “java.lang.RuntimeException: Unable to start activity ComponentInfo.” and finds a question on StackOverflow on the topic. The top answer says the problem is a missed dot in the name of the of the activity in the xml.
3) hypothesis - Is the application missing a dot in the activity name?; the participant wonders if the application has the same error as the top answer on stack overflow.

5) the participant checks the Android manifest file and determines that the period in the activity name is not the issue.

2) the participant comments that Android runtime exceptions don’t usually show the problematic line in the stack trace. He also comments that the fix probably only takes a couple lines of change once you know the cause (comments that it doesn’t take that much time), but the difficult part is finding the cause.
4) the participant puts logging statements in the onCreate method of LNotificationActivity.but seems to give up on the idea and goes back to Googling the exception.
7) the participant Googles “java.lang.IllegalStateException: The specified child already has a parent” and finds another StackOverflow question. The top answer states when you override OnCreateView, if the method has if(view != null) {return view;} then the developer should remove the return view statement.
8) The participant explains that there were two exceptions in the stack trace and he just Googled the second one. *I’m assuming that the participant stopped the logging earlier because he noticed there was another exception message to Google.*

5) the participant checks the code and notices that the HeadUpFragment's onCreateView method does not contain the return view statement.
7) the participant goes back to reading the comments on the answer. The last line of the comment contains a fix to the directive in the task: **onCreateView if using inflate(), then pass False to inflate()'s third argument**
3) what does the inflate method do? the participant decides that he should better understand the inflate method and so he decides to look up the method. (8:03)
7) the participant googles the inflate method and selects the first link, which is a blog article on the inflate method.
7) the participant scrolls downs and sees a code example where the inflate method is called with the last parameter set to true
3) hypothesis: the last argument for the inflate method should be true
    4) the participant adds a true parameter to the end of the inflate method in HeadsUpNotificationFragment
    5) the participant reruns the application and sees the same error message
7) the participant Googled the error message "The specified child already has a parent. You must call removeView() on the child's parent first."
8) the participant comments that the participant is trying to find a quick fix and wants to see if there is something similar to the code in the webpages
7) the top answer on the page says that the original poster of the question should change his onCreateView method from what the original poster had to an inflate method with false at the end
3) hypothesis: should the last parameter be false?.
    4) the participant changes the last parameter to false
    4) the participant tests the application and determines that it is working correctly (12:33)

Participant 8, Task 6, Fault 50

8) the participant reads the prompt to understand the problem
7) the participant runs the application to see if it produces any error message
8) the participant notices that no error message is produced and the icons are not appearing in the top right
3) Hypothesis: Is there something obviously wrong with the code?
    2) Some how the participant thinks that the error is likely in the LNotfiicationActivity file
    7) the participant scrolls through the code in LNotificationActivity
    5) after scrolling up and down through the file, the participant doesn't notice anything and gives up
7) the participant Googles "can't see setting menu Android", but then changes the search when the participant sees the results to "site:stackoverflow.com settings menu not visible android"

2) the participant seems to trust the answers at StackOverflow compared to other options
7) the participant clicks on the first answer and reads that the original poster of the question should override the onCreateOptionsMenu
The onCreateOptionsMenu should be

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  MenuInflater inflater = getMenuInflater();
  Inflater.inflate(R.menu.game_menu, menu);
  return true;
}
```

3) hypothesis: should the application override the onCreateOptionsMenu?

7) the participant looks through the LNotificationActivity and notices that the method is not overridden in the class
4) the participant overrides the method in the activity in a way that is similar to the example.
7) while doing the above, the participant stops to click the link the in StackOverflow answer to read through the Android documentation on the question.
4) the participant changes R.menu.game_menu to R.menu.call_planning which the participant found in the menu folder (same folder as where the example said to find the xml)
4) the user tested the application and found that the menu appeared. (20:15)

Participant 8, Task 2, Fault 29

8) the participant read through the prompt to understand the problem
7) the participant then runs the application to see if it provides any information on the problem
7) the participant check through the warning messages at the log and seems confused about what is actually the problem
7) after a comment by the researcher, the participant goes to HeadsUpNotificationFragment and reads the warning message: **This fragment should provide a default constructor (a public constructor with no arguments) (com.example.android.lnotifications.HeadsUpsNotificationFragmment)**

From the Fragment documentation:
Every fragment must have an empty constructor, so it can be instantiated when storing it's activity's state. It is strongly recommended that subclasses do not have other constructors with parameters, since these constructors will not be called when the fragment is re-instantiated; instead, arguments can be supplied by the caller with setArguments and later retrieved by the Fragment with getArguments(). (24:09)

7) the participant Google's "set fragment Android" to better understand the problem, but then quickly changes the results to "set fragment bundle Android".
7) the participant finds an example of how to add arguments to a Bundle and how to retrieve arguments from a bundle in the StackOverflow question.
7) the participant goes back to reading the warning message
7) the participant Googles "set arguments fragment" from the error message
2) the participant seems to have a vague idea how to implement the code correctly from the warning message but doesn't know how to do it exactly so they have to look up more information on the methods and classes in the warning message
7) the participant skips the first link and selects the StackOverflow question on how to use setArguments() and getArguements() in Fragments
7) the top answer to the question seems unhelpful so the participant goes back to the code
4) the participant tries to implement his understanding of what the warning message says to do - use a default constructor and pass the argument with a bundle

5) the participant removes the parameter from the constructor call
5) the participant then adds code to create a Bundle after the constructor call
7) the participant Googles "make a new bundle" and changes it to "make a new Bundle Android" after seeing most of the result were not Android related
7) the participant selects the first option: a StackOverflow question on creating bundle and sending over to new activity
5) the participant adds code to attach the Bundle to the fragment
2) the participant realizes that the participant needs to put the GregorianCalendar into the bundle but isn't sure exactly how to do that because there isn't a method for that type.
8) the participant googles "put Class in Bundle Android and reads the StackOverflow answer to user serializable
5) the participant adds a putSerializable line to the Bundle setup
5) the participant runs the application and sees that it crashes (31:40)
7) the participant Googles a stack overflow question on bundle getSerializable to figure out how to pull out the information from the bundle
4) the participant copies an example for how to get the serializable object out of a Bundle from StackOverflow into the code and then changes the data types in the example to fit the situation
2) the participant comments that copying the code and changes the datatypes is usually the faster way to fix these types of errors.
4) the participant adds a try catch block around the code to fix a compile error while changing the copied code
4) the participant runs the application again without the code crashing but the time text does not appear at the bottom of the fragment
7) the participant reads through the log messages from the emulator and finds a line about the a null pointer exception on the serializable call, which likely was created by the participant's try-catch statement

7) the participant looks back at the code and realizes that the participant forgot to change the key to extract the serializable from the Bundle, so there was nothing with the last used name in the Bundle.
4) the participant changes the key to match the key the participant used when creating the Bundle
4) the participant checks the application again but still doesn't have the time appear
7) the participant goes back to reading how the participant set up the Bundle
4) the participant decides to add a logging statement in the catch part of the try-catch block the participant added to make sure the application is taking the same route the participant expected.
4) after confirm the statement was printed, the participant added debugging statements in the try part of the application to determine which line the application is failing in the try block.
2) the lack of print statements cause the participant to assume the first line failed: the line with the getSerializable call
7) the participant Googles the failing method call with the error message and clicks on a Stack Overflow answer
7) the participant confirmed that the participant was using a similar method to the question asker. The answer seemed to imply that the question asker was using the wrong way to get the Bundle at the failing location, but the participant's code was failing when the Bundle is passed by an argument.
7) the participant decides to Google about the bundle parameter name "passing Bundle via savedInstance state" and selects another StackOverflow question
7) the participant reads the question where the StackOverflow question asker uses a different Bundle than the parameter (one obtained by getArguments) before passing the parameter to the super class.
2) the participant thinks that the wrong Bundle is being passed as the parameter
4) the participant decides to implement the method in the same way as the question and comments that the participant "isn't sure why they make you do this".
4) the participant starts the application
7) the participant reads through the StackOverflow answer, *seemingly to make sure that the answer didn't say that part of the question was the problem*
4) the participant comes back to the application and determines that the application works
2) the participant comments that the solution was unintuitive - "why do they make you do this?"

Participant 7, Task 5, Fault 63
8) Participant reads through the prompt to understand the problem.
7) the applicant tests out the application see how it is failing and see the stack trace

7) the participant test out the case where the application is working to see what the final application should do
7) the participant then tests the application by first going through the passing sequence and then the failing sequence and determines that the application still fails
7) the participant tries to go through he failing sequence again and notices that there is a problem with the instructions for the question (it has an extra step before the application fails that is not occurring)
7) the participant reads through the stack trace to find locations in the current project where the application is failing
7) the participant notices that the stack trace only mentions one location in the current project,
2) the information above causes the participant to assumes that the bug is only in that location
7) the participant clicks on the source in the bottom of the page
7) the participant reads through the method which contains the failing line
7) the participants reads through the code above the failing line
2) the participant deduces that the error may also be in the HeadsUpNofificationFragment since that is involved in the failure process
7) the participant reads through the top of the HeadsUpNoficiationFragment file
3) Is the setRetainInstance parameter wrong in the HeadsUpFragment?
    4) the participant changes the parameter from true to false
    2) the participant comments that this fix may work because the error message states that the fragment is already active. Thus the problem may be that the fragment isn't closing correctly. The participant mentions that the problem may be with how the Bundle is used but the participant wants to test the option first.
    7) the participant Googles onReationInstance but gives up when the participant doesn't quickly see it on the Fragment documentation page
    2) *the participant seems to think that testing the application would be faster than reading through the documentation*
    7) the participant tests the change
    5) the participant finds that the application still crashes
7) the participant goes back to reading through the method with the failing method call
7) then the participant scrolls through the main activity to see if there are any problems
2) the participant mentions that the participant still thinks that there may be something wrong with how the fragments are being retained when they shouldn't be.
2) the participant then remarks that the participant isn't sure what the problem is after searching through the code
8) the participant then mentions that the participant usually Googles the error when the participant isn't sure what else to do
7) the participant Googles java.lang.illegalStateException: Fragment already active - the top error message of the stack trace
7) the participant reads the top answer which says that the in Android, **a developer cannot use setArguments twice on the same Fragment**

3) is the problem that setArguments is used incorrectly (specifically called twice like in the example)?

4) the participant comments out the setArguments statement in ChangeColorFragment's onViewCreated method and replaces the Bundle instantiation with a call to getArguments()

7) the participant tests the application and finds that it is still failing

5) the participant is happy that that there is a new stack trace. The participant is happy that the change is having an affect on the program, which is better than making a change with no noticeable effect. The participant comments that changing exceptions mean that you are more likely to notice what is going on. (19:42)

4) the participant changes the setRetainInstance from false back to true, realizing that the participant had forgot to do that earlier

3) Is the problem now that the headsUpFragment bundle is not instantiated before it is called?

2) the participant mentions that the participant is looking for the creation of the fragment, so the participant can set default arguments to fix the null pointer call from getArguments()

2) the participant mentions that the participant is used to using adaptors to hold tabs but this example is not using an adaptor

4) the participant adds the line headsUpNotificiationFragment.setArguments(new Bundle()) to the onCreate method in LNotificationActivity

7) the participant tests out the application and finds that the application is not crashing but the button color is not changing

4) the participant starts to remove the argsBundle.getInt line because it contains the the parameter Color.LTGRAY but then realizes that the second argument is the value to return if the key sent in the first argument is not found

4) the participant adds a debug line under the line the participant was about to delete to test which value the getInt method is returning.

7) the participant runs the test and the output for the color is -3355444

2) the participant doesn't seem to understand the output

7) the participant tests if the output will print every time the button is clicked, and finds that the output only prints once, meaning onViewCreated is only called once for the fragment

2) the participant comments that his understanding of when onViewCreated is called seems to be off

7) the participant reads through the activity code to see how the fragments are changing on tab switch

2) the participant notices that the tabs are all initialized at the beginning and then the fragments are saved until needed

2) the participant realizes that the fragments are not calling onViewCreated when they are switched, since they already were created, but the participant is not sure what is called when the fragments are switched.
7) the participant goes back to the previous StackOverflow page looking for possible clues how to solve the problem
7) the participant looks back at the code to see if some of the answers are options in this case but decides the answers don't apply
7) the participant goes back to looking through the code to find a hint for the method called when fragments are changed
7) the participant decides to Google "FramgentTransaction show which function is called in the fragment" and select the first stack overflow answer
7) the participant reads the accepted answer about onStart() being called when the fragment is shown
3) Should the button color be set in the onStart method?
    4) the participant overrides the onStart method and implements code to set the button color to the arguments value
    5) the participant tests the application and finds that it the onStart() method isn't being called when the active fragment changes
7) the participant goes back to reading through the StackOverflow questions for ideas (43:26)
7) the participant looks for possible methods to call on the view
2) the participant mentions that the button color change needs to be added to the view
3) should the button color change be added to the view?
    4) the participant realizes that the findViewById call on the button was not in the onStart method like it is in the onViewCreated so the participant adds the line to the onStart method
    5) the participant tests the application and finds that the change doesn't fix the problem
7) the participant Googles reinitialize fragment view and clicks on the first stackOverflow question about reinitializing the back stack on fragment change
2) the participant mentions that the view is loaded after the onStart
2) to the participant mentions that the participant needs to change the code to do something different in the onTabSelected method in the main activity
7) the participant goes back to the onStart StackOverflow answer and clicks the link to the Android lifecycle webpage.
2) the participant notices that onStart executes when the parent activity is started
7) the participant Googles "view not updated in fragment onStart" and opens the first two StackOverflow answers in different tabs
7) the participant reads through the answers on both tabs but doesn't seem to find what the participant is looking for

7) the participant searches again for the same question and opens up the third link
7) the participant finds an answer about using onViewStateRestored
3) is the problem the onViewStateRestored method?
    4) the participant overrides onViewStateRestored with a debug message and adds a debug message to the onStart method. The participant also adds a debug statement to onViewCreated
    5) the participant finds that switching tabs doesn't cause the other methods to be called
2) the participant states that the show method is causing the tab switching. The participant thinks that onShow should call onStart but the onStart method is not being called in this case
7) the participant tries to Google a question about which method is called with show() ("FragmentTransaction show which function is called in the fragment") and clicks on the first StackOverflow question.
7) the participant quickly scrolls through the answers and doesn't see anything that obviously applies in his case.
7) the participant then slowly scrolls up and decides to copy the answer which covers the setUserVisibleHint method
3) is the problem the setUserVisibleHint method?
    4) the participant overrides the method in his code.
    7) the participant tests the application and doesn't notice a change
    4) the participant adds a debug statement to the method
    5) the participant tests the application again and determines that the method is not being called (1:01:28) so the participant deletes the method
7) the participant looks up the Android documentation on the show method
3) Is the fragment not being recreated because of the show method?
    4) the participant removes the show call so the fragments are added every time they are clicked
    7) the participant tests the application and gets a fragment already added exception
    4) the participant restores the show call in the activity
    7) the participant goes back online and finds a stackoverflow answer about using remove when the fragment is done
    4) the participant tries to add code to remove the old fragment and add the new fragment but does so incorrectly, where the participant is removing the current fragment and then adding it
    7) the participant tests the application and finds that the tabs are now blank
    5) the participant undoes the changes to the onTabSelected method
5) the participant decides that the participant doesn't know the basics of the fragment lifecycle and decides to that the fix would take too long. The

participant decides to end the test and decides that it would take about another hour to fix the application.

Participant 9, Task 4, Fault 13
7) the participant reads through the prompt to understand the problem
2) the participant mentions that the prompt makes the participant think the application problem has to do with a Toast popup
7) the participant runs the application to see the problem where the app crashes
7) the participant looks that the top of the application to see the error message
7) the participant goes to the failing line with the null pointer reference in the OtherMetadataFragement
7) the participant scrolls through the rest of the file to see if there are any problems with the activity
7) the participant starts to investigate the getActivity method to see where it was defined (see if the activity call is implemented incorrectly or if it is part of the framework)
2) the participant notices that other getActivity calls occur in the file
3) “Why is this getActivity call not working?” “Is the Activity not created yet?”
3) Does the activity work if the failing getActivity call is removed?
4) the participant changes the line to remove the failing getActivity call but leaves another call to getActivity that the participant didn’t notice.
7) the participant tests the application and notices that it fails
7) the participant notices the other getActivity call in the line caused the application to fail
5) *the participant seems unsure of what he could replace the other parameter with and so tries a different approach*
2) the participant states that the participant is unsure why the getActivity call is returning null
3) Why an getActivity return null?
7) participant decides to first look through the getActivity code to see if there is an obvious problem with how the fragments are being registered to the activity for convenience purposes
7) the participant notices the special handling in the Activity for onTabSelected
7) the participant searches online for “getActivity null” to find cases where the activity could be null
7) the participant mentions that the participant usually finds StackOverflow helpful
7) the answer states that the problem is due to calling the method after the activity is deleted
3) hypothesis: is the problem in the code that the activity has been deleted before it is called?
7) the participant decides to first scroll through the OtherMetadataFragment to find any odd pieces of code that may provide clues

7) the participant searches through OtherMetadataFragment and HeadsUpNotificationFragment to determine that they both don't implement onAttach
4) the participant overrides the method onAttach in the OtherMetadataFragment and saves the activity in an object(16:00)
4) the participant wants to replace all calls to getActivity with the saved Activity but is having difficulty replacing the calls in an easy way
2) the participant mentions that the participant isn't sure if onAttach is called before the other methods which need the Activity
2) the participant questions if the participant should remove the call to the displayActivityTitle method
4) the participant replaces the getActivity calls in the failing line with the Activity reference saved in onAttach
4) the participant realizes that the participant tried to call the super method wrong in the onAttach method after receiving a compile error, the participant fixed the problem by looking back at the StackOverflow answer
7) the participant generated another exception
3) is the emulator using the latest version? The participant is confused if the emulator is still running an old version
7) the participant looks at the stack trace and then scrolls back to OtherMetadataFragment

3) is the problem the call to displayActivityTitle in the Activity, when it should be in the correct place in the OtherMetadataFragment
3) the participant asks the researcher if the participant can move the displayActivityTitle call from the Activity to the OtherMetadataFragment
4) the participant adds the call to the end of onViewCreated and removes the call from the Activity
7) the participant tests the application and finds that the program works (26:00)

Participant 9, Task 6, Fault 50
7) the participant reads the prompt to understand the problem
2) the participant thinks that the problem is likely related to the ActionBar because from experience, the participant thinks that options can be added there
7) the participant tests the application and finds that the icons are not appearing
7) the participant looks through the Activity quickly and finds the declaration of an ActionBar
2) the participant mentions that a member function of another class was called in a weird place in the last task, so this task may have a similar problem
7) the participant looks through the Activity code to see what the ActionBar does
2) the participant comments that the ActionBar is used to set up the tabs themselves
7) the participant looks through HeadsUpNotficationFragment to see if the participant notices any problems

2) the participant comments that the participant would have looked for the action handling code for the buttons if they were supposed to do anything
7) the participant looks through the layouts to see if they contain the icons
3) is the the title of the application is part of the Activity and not the Fragment?
    7) the participant looks for the application title and looks for where the title is used by first find the title in string.xml and then right clicking the string to search for usages
    7) the participant finds the title being used in the onCreate method of the LNotficiationActivity (35:35)
2) the participant mentions that the participant thinks that the code needs to changed in the LNotificationActivity
2) the participant mentions that the participant is used to an older version of Android but thinks that that the change should be in the ActionBar in this version
7) the participant Googles “android add menu button to title bar” and clicks on the link about adding the overflow menu to the title bar.
7) the participant scrolls through the answers and finds a comment that you may want to explore the options of the ActionBar.
7) the participant decides to click a link which takes the participant to an Android page about the app bar, which the participant comments that the picture looks similar to what the participant is trying to do.
7) the participant clicks the link about adding an up button to the bar and reads through the web page instructions (39:22)
7) the participant concludes that the answer seems to be something so specific that it covers something already provided in this case.
7) the participant next clicks on the link about adding and handling actions.
7) the participant learns about how to respond to different action presses in the options menu but not how to add the menu
7) the top of the file, states that the menu resource should be defined in an xml file.
7) the participant looks at the example for the naming scheme to determine where the menu might be or where to add the menu to the application
2) the participant mentions that the participant is still unsure how to automatically display the action with the action bar
7) the participant goes back a page and then clicks on the link about setting up the action bar
7) the participant finds that the page recommends using AppCompactActivity to get backwards compatibility
2) the participant mentions that the participant is confused about the compatability issues but decides to try the backwards compatibility method
7) the participant looks through the possible actionBar methods to find another option
7) the participant then decides to go through the drawables to look for the two different icons on the picture
7) the participant doesn’t find it but the researcher mentions that the icons are default Android icons

7) the participant instead looks for a menu definition and finds it in the menu folder.
2) the participant comments that the participant isn't sure this is the right menu but the participant thinks the xml file in the menu folder looks right
2) the participant comments that the participant doesn't see the showAsAction information defined like in the example
3) is the problem that call_planning.xml does not have showAsAction defined in the file
    4) the participant adds a showAsAction to the file
    4) the participant notices that there is a build error because showAsAction is defined twice, so the participant removes the added line
    5) the participant runs the application and determines that the menu still doesn't show and thus showAsAction is not the problem
7) the participant searches for "action bar not showing icon" and clicks on the second link, because the first one mentions app compact, which the participant has determined is the old version
7) the participant reads through the answers and finds that one of the down voted answers say that there is no title defined for the menu item
3) the problem the icon title?
    7) the participant notices that AndroidStudio recommends defining the icon title in the strings.xml file and not hard coding it.
    4) the participant adds the title string to strings.xml and replaces the hard coded strings with references in the call_planning.xml file (1:00:37)
    5) the participant tries to run the application and finds that the icons still don't change
    2) the participant comments that the change was something valuable to do (assuming because of the AndroidStudio warning)
7) another answer on the question says that the xml needs another namespace for the menu (not directly but that is what the participants says the participant got from it)
3) is the missing namespace the problem?
    2) the participant comments that everyone seems to use the compatibility library in this case
    4) the participant adds the namespace to the file
    5) the participant tries to add a showAsAction for the namespace but gets a compiler error
    4) the participant removes the change
7) the participant goes back to the Google search and skips many results because of the date
7) the participant goes to the code and tries to find optional method calls for the action bar
7) the participant doesn't find anything and so the participant searches for "android lollipop actionbar icon"
7) the participant skips options that mention the app compact version and chooses a question that seems to be up to date for the participant

7) the participant reads an answer about using getSupportAction bar; the participant makes a comment about it being easy enough to try
3) does the application need to use getSupprtActionBar?
    4) the participant tries to add a getSupportActionBar and gets an error that the main class needs to extend AppCompactActivity.
    5) the participant tries to change the extended class to AppCompactActivity but that class is not available in the current API
    4) the participant undoes his changes
3) is the problem the application theme
    7) the participant tries to find the theme in the code and then looks online how to find it.
7) the participant search for linear layout action bar after looking it up in the main xml
7) the participant clicks on a vogella link about the Android action bar
2) the participant comments that the participant has used Vogella's site before
7) the participant finds that the page recommends to **override the onCreateOptionsMenu method**
2) the participant comments that the participant isn't sure of the menu file that the participant should use at the moment
3) is the problem that the activity doesn't have the onCreateOptionsMenu?
    4) the participant implements the onCreateOptionsMenu
    7) the participant tries the application and finds that the icons appear

Participant 10, Task 7, Fault 12
7) the participant reads the prompt to understand the problem
7) the participant runs the application to see the failure and the stack trace
3) hypothesis: is the problem a layout issue?
    7) the participant concludes that the error means that the application has a layout issue, due to the fact that the participant has ran into the problem before
    7) the participant starts to look through the xml files in the application for a duplicate ID
    2) the participant thinks the layouts look fine and comments that a duplicate may be added in the code
    7) the participant looks through the activity for a duplicate addition but doesn't find anything
    7) the participant looks through the stack trace again to see if the participant can use it to determine the error location
    2) the participant comments that the problem is likely in the HeadsUpFragment because the application fails at set up and the participant doesn't notice a problem in the Activity
    7) the participant looks through the HeadsUpFragment file
    2) the participant comments that the participant knows the problem is a layout issue for sure - "clearly the problem is a layout issue, I know that for sure"

7) the participant goes back to looking through the code the participant previously checked
7) the participant switched from the xml view to the design view for the xml files to find an unable to render error
7) the participant finds a render problem in the fragment_other_metadata.xml and investigates the file.
2) the participant knows that ScrollView only accepts one child, which may be the problem
3) is the problem the ScrollView in the file?
    7) the participant reads through the file but doesn't find anything
7) the participant goes back to looking through the other xml files
7) after checking the xml file, the participant goes back to trying to get information from the activity and the error stack
2) the participant comments that the Android debugger commonly slows the participant down when developing (mentions that it is mainly useful for large code), the participant mentions that the participant usually has an idea where the bugs are in the code, but the participant mentions that the participant doesn't know in this case, so the participant plans to use the debugger
3) is the problem related to the theme?
    7) the participant tries to find the theme
    2) the participant comments that the problem may be the theme, because the no action bar theme could cause problems with an action bar
    5) the participant finds the theme and determines that the theme is not the problem since it is not the no action bar theme.
7) the participant decides to add a break point after the tabs are initialized to determine if they are a problem
7) the participant tries to use the debugger to find out where the application is crashing
7) the participant gets lost when debugging through the Android framework code (since the docs aren't set up and the debugger is referencing stub methods) and the participant concludes that the OtherMetadataFragment is causing the crash
2) is the application showing the HeadsUpFragment by default?
7) the participant looks up "ActionBar Tabs" and clicks on a github link to ActionBar Tab with Fragments
3) is the problem that the action bar is missing the selected tab function?
    4) the participant adds actionBar.selectedTab(HeadsUpFragment) and tests the application
    5) the participant sees that the error message is the same
    2) the research questions the participant why the participant put the line in that location, since the participant seemed to conclude that the application failed before that line with the debugging. The participant

mentions that the participant forgot that the application wouldn't reach that line.

7) the participant sets another break point for when the visibility tab's tabListener is created

7) the participant tries to filter out the debug statement the participant found in the application

7) the participant runs the debugger and finds that the application gets through the method that the participant thought was originally failing but the the participant gets lost in the Android method stubs (23:03)

7) the participant goes back to looking through the HeadsUpFragment

2) the participant mentions that onCreate clearly goes through to the end so the problem is likely in a fragment

2) the participant considers putting a break point in onCreateView but instead puts the break point in onViewCreated because the participant doesn't see any problem in onCreateView, the participant also mentions that the break point should provide insight into if the onCreateView method has a problem, because if the application crashed before reaching the break point, the participant would know that the onCreateView method worked

7) the participant finds that the application crashes before reaching onViewCreated

7) the participant goes to OtherMetdataFragment because it is the Fragment that is added

2) the participant comments that the current approach is a hunch, but the participant thinks that the program gets through adding the First tab successfully but the application seems to crash on the last tab (states that the problem is not in the first tab)

2) the participant comments that the participant didn't write the code, which makes the process difficult

3) is the problem in a fragment?

4) the participant comments out all the fragment initialization code and the code to add the fragment to the activity to see if the application can start successfully - the participant tries to debug with components

7) the participant finds that the application works without the tabs

5) the participant concludes that the problem is definitely in a Fragment

2) the participant states that the participant isn't sure where to put the break point, and so the debugger isn't much help in this case, and based on the participant's experience, the participant thinks that the problem is a layout issue

7) the participant looks through the Activity code and notices that there is not a commit call after the replace call

3) should there be a commit call after the replace call

7) the participant looks up the FragmentTransaction to see if it needs to be committed

4) the participant adds a commit call after the replace call

7) the participant tests the application which crashes
4) the participant removes the commit call
7) the participant goes back to looking through the OtherMetadataFragment and then the participant looks through other fragments and then the stack trace to see if replace is the error
7) the participant then tries to look at the replace documentation
7) the participant reads through the AndroidStudio method documentation and concludes that replace does not call commit
2) the participant mentions that the participant isn't sure about where the call to commit is called
2) the participant mentions that the participant still thinks the problem is due to the layout
5) the participant gives up look for commit

7) the participant goes back to the xml files to look for the layout problem

2) the participant says that the participant is too confused and wants to give up
2) the participant mentions that the participant would have to do a lot of Googling to fix the problem
2) the researcher asks the participant why the participant hasn't tried that strategy
2) the participant comments that the participant was biased by previous knowledge of a similar error
7) the participant Googles the error message and clicks on the first StackOverflow question
7) the participant finds that the **inflator method should have the parameter of false at the end**
3) hypothesis: is the problem that the inflator method doesn't have the false parameter
4) the participant adds false to all the inflator method methods
7) the participant tests the application and finds that it works

Participant 2, Task 1, Fault 31
7) the participant reads through the prompt to understand the problem
7) the participant tries to step through the task in the application
7) the participant looks through the application to find the problem, specifically the set meeting button listener
3) hypothesis: is the fault location the button listener
4) the participant changes part of the button input
7) the participant tests the application and confirms that the button listener is causing the erroneous message
2) the participant comments that the participant needs to fix this area but now needs to figure out how to get the correct data there (33:08)
7) the participant starts to look at the meeting fragment xml to get ideas for how to access the time from the time buttons
2) the participant comments that the participant understand how the buttons are initialized but still doesn't understand how to access the buttons

3) how can the time selection buttons be accessed when the set meeting button is clicked?

3) can the time selection fragment be referenced in the button listener?

5) the participant tries to access the tsf variable in the button listener but finds that the button is out of scope

7) the participant reads through the meeting fragment and also reads through the fragment_meeting.xml (39:30)

2) the participant deduces that the FragmentTransaction is a local variable

4) the participant tries to look for a get method using autocomplete to get the buttons for the listener but doesn't find anything helpful

4) the participant tries to use a toString on the layout file but receives a compiler error

2) the participant comments that the participant is trying to figure out how to get the buttons.

2) the participant comments that the participant knows the buttons are being added to the FragmentTransaction, and is trying to figure out if the buttons can be extracted from there.

7) the participant googles "how to get elements resID Android" and selects an Android documentation page on accessing resources in code

7) the participant finds an example to access the element using findViewById

7) the participant goes back to looking through get autocomplete options in the code

7) the participant goes back to looking through the meeting xml layout

7) the participant looks up how to use print statements in Android for debugging

4) the participant adds a line to print out the TimeSelectFragment Id

7) the participant runs the application in debug mode

7) the participant Googles "redid vs getid android" but doesn't click on any of the links

7) the participant finds that the id of the fragment transaction is the same for both buttons, so the participant can't use the fragment transaction to access the different buttons

4) the participant tries to create each button separately in hopes of being able to distinguish the two buttons in code

4) the participant then tries to access the code using the different time selection fragments and comments that the participant needs the resID, even if the participant has the button.

2) the participant gets confused for a bit about the final requirement for accessing the button in the listener

4) the participant adds the final keyword to the time selection fragment

8) the researcher calls the study (in hopes that the participant will have time to complete the second task)

2) the participant comments that the participant is having difficulty because the participant doesn't understand the Android way of doing things.

2) the participant estimates that it would take another 8 hours to fix (although this number was planted by the researcher so it may be invalid) (1:03:14)

Participant 2, Task 7, Fault 12
7) the participant reads through the prompt to understand the problem
2) the participant summaries that the program has a runtime exception and the participant must make the test cases pass
7) the participant runs the application to see the result
7) the participant reads through the error message
2) the participant concludes that the error message sounds like it came from the onCreateView method or a similar method
3) hypothesis: is the problem the onCreateView method?
7) the participant glaces through the onCreateView method of LNotificationActivity
7) the participant goes back to error message and confirms that the error message came from LNotificationActivity
3) “which child already has a parent?” What is the error message talking about?
2) based on the error message the participant concludes that somewhere an extra child is being added and that the participant needs to add a removeView call
3) hypothesis: are the fragments the child that the error message is referring to
    7) the participant scrolls through the Activity file but doesn’t notice anything obvious
    7) the participant sets a debug statement in the onCreateView and determines that the problem is not occurring in that method
7) the participant tries to search for the Class ComponentInfo, which was referenced in the error message
7) the participant Googles how to find a string in AndroidStudio project
7) the participant uses the find command to find the ComponentInfo but doesn’t get any results
3) where does the application go after onCreate?
    7) the participant debugs the application to the end of the onCreate method
    7) the participant adds another breakpoint at the end of createNotification in HeadsUpFragment
    7) the participant learns that the breakpoint is not executed before the application crashes
7) the participant decides to go back to searching for the ComponentInfo class
7) the participant then searches for view
4) the participant tries to add a removeView call at the beginning of onCreate in the Activity by looking through autocomplete options but the participant doesn’t find what the participant is looking for
7) the participant also tries to look through the autocomplete options for se but then deletes the line after looking through the results
7) the participant Googles for an example for “onCreate Activity Android” and clicks on the documentation page for the Android lifecycle

7) the participant finds an example of the onCreate method on the page and reads through it
7) the participant looks through HeadsUpNotification for a view that the participant may remove with removeView
7) the participant decides to look into the onViewCreated method in depth to see if a view may be added there
7) the participant runs the application again with the debugger
8) the participant stops the task
8) the participant predicts that the task will take two hours more to solve

Participant 3, Task 6, Fault 50
7) the participant reads through the prompt to understand the problem
7) the participant runs the application and finds that the icons don't appear
7) the participant looks for a problem in the Activity file, mainly looking at how the ActionBar is being set up
7) the participant then checks other tabs in the emulator *to make sure the problem isn't related to a fragment*
3) hypothesis: "do the other menu options work?"
    7) the participant clicks through the different tabs and finds that the tab switching works but icons do not appear for different tasks
7) the participant starts to look through the HeadsUpFragment code (13:40)
7) the participant then goes to through the xml files to see if the participant notices any problems
7) the participant wants to look back at the previous project but the researcher says that is not an option
7) the participant Google's "ActionBar adding a menu with icons"
7) the participant clicks on the Android menu documentation and then clicks back without reading the page
7) the participant also clicks on a StackOverflow question about showing icons in the ActionBar overflow menu but also quickly exits the page
7) the participant then goes back and clicks on an Android documentation page about adding actions
7) the participant reads about adding the items in xml and assumes that is related to the problem
3) hypothesis: is the problem a missing menu xml?
    4) the participant starts to create a new menu in the xml and pastes the example code from the Android documentation site
    7) the participant questions where the icons are in the project
    4) the participant changes the code to work in the current project using drawables found in the drawable folder
    2) the participant comments that the participant should first see if the code causes icons to appear, then the participant can worry about using the correct icons

2) the participant realizes that the participant needs to add the menu in the Java files somewhere
7) the participant tries to read through the page but doesn't find what the participant is looking for
2) the participant comments that the participant has forgotten how to add the menu
7) the participant starts looking through the fragments code to find how to add the menu to a fragment but doesn't find anything useful in any of the fragments
7) the participant decides to Google search "how to add a menu to android app"
7) the participant reads through sample code on a code sharing website
2) the participant notices that the menu is added using the inflator.inflate method
7) the participant looks through the code and notices that that the fragment already has an xml defined in the inflator.inflate call
4) the participant deletes the old menu xml and replaces it with the participant's defined xml
7) the participant tests the application and determines that the icons are not appearing
7) the participant goes back to the xml file the participant wrote to make sure that there wasn't any errors
7) the participant then checks what icons the participant specified should look like
8) the researcher tells the participant to look at the call_planning.xml
2) the participant notices that the xml file defines the icons but the participant can't see the icons in the drawable
3) are the icons missing?
7) the researcher says that the icons missing is probably not the problem
7) the participant runs the application again and seem confused why the menu is not appearing
7) the participant reads through the HeadsUpNotificationFragment code
7) the participant reads through the layout files
4) the participant tries a different icon in the participant defined menu xml file
7) the participant tests the application again and finds that the icons still do not appear (30:03)
7) the participant states that some icons appear on the visibility tab so the participant looks into the visibility tab but doesn't find anything
7) the participant scrolls through a couple of different project files
7) the participant looks back at the Android documentation page for more clues but doesn't seem to get anything
7) the participant searches for "android add menu button to action bar tab"
7) the participant seems to find the same developer pages before but the participant scrolls through it looking for clues
7) the participant looks through the xml files to see if anything is missing
7) the participant Google's "how to add a menu bar to a fragment in android"

7) the participant finds that the top stackoverflow page says that the participant should call the super method in the onCreate method and also that **onCreateOptions menu should be overridden to add the menu items**
3) hypothesis: is the problem the super call in the onCreate method?
    5) the participant checks the fragment and finds that the onCreate super method is called correctly (36:56)
7) the participant goes back to the StackOverflow answer and sees the onCreateOptionsMenu should be overridden
3) hypothesis: does the application need a correct implementation of onCreateOptionsMenu?
    4) the participant copies and pastes the method example into the code
    4) the participant add setHasOptionsMenu(true) to onCreate
    7) the participant finds that the icons the participant defined now show up
4) the participant changes the menu to the call_planning menu
7) the participant runs the application and finds that if works
2) the participant comments that the the participant was confused with the context menu and the options menu, and that is why the participant had so much difficulty
2) the participant comments that because you can't see the icons in the project, there is the possibility they are causing the problem until you replace them with something else

Participant 3, Task 4, Fault 13
7) the participant reads through the prompt to understand the problem(41:15)
7) the researcher demonstrates the crash
7) the research demonstrates the failing test case
3) the participant asks if the test case is demonstrating the same error as the crash
7) the participant reads through the stack trace (42:27)
7) the participant finds that the activity is null in OtherMetadataFragment
7) the participant starts to look through OtherMetadataFragment to look at the failing line
4) the participant writes a getActivity call before the line along with a print out of the results
7) the participant debugs the application and confirms that getActivity is returning null
4) the participant adds a comment that getActivity is null at that line
7) the participant first looks for getActivity uses in the OthersFragment and then for references to the OthersFragment in the Activity
7) the participant then searches online for "android getActivity in fragment returns null" and clicks on the first StackOverflow question and reads through the answers to figure out possible reasons for the failure
7) the participant finds an answer about saving the activity reference by overriding the onAttach method
3) is not saving the activity reference the problem?

4) the participant overrides onAttach and changes the code to use a saved activity reference
7) the participant runs the application and finds that the application still crashes
5) the participant debugs the application and finds that the reference is null
2) the participant comments that the participant isn't an expert in Fragments, the participant only understands the basics
7) the participant searches online for getActivity() in Fragment returns null
7) the participant clicks on the second Stackoverflow answer because the participant had already clicked on the first one
7) the participant reads through the answers and find that the accepted answer talks about using onAttach but also mentions the onDetach method
3) is the application using onDetach correctly?
7) the participant scrolls through the OtherMetadataFragment
3) where is the displayActivityTitle called?
7) the participant finds the activity call in LNotificiationActivity
3) can I pass the activity in the method call?
4) the participant changes the code to pass an activity to the displayActivityTitle method
7) the participant realizes that the FragmentTabListener doesn't have the activity
4) the participant changes the FragmentTabListener constructor to take an activity and then passes the activity to the FragmentTabListener as an argument.
7) the participant tests the application and determines that the application works and doesn't crash

Participant 3, Task 5, Fault 63
7) the participant reads through the prompt to understand the problem
7) the participant executes the application to see the crash
7) the participant looks through the stack trace and reads that the error occurred in the changeColorFragment.
7) the participant then starts looking through the changeColorFragment and reads through the onClick code in the onViewCreated
7) the participant runs the application again to review the problem
7) the participant googles "Fragment already active" and clicks on the first StackOverflow answer
7) the participant read through the response that you can't use setArguments again on the same Fragment (**this is indirectly the directive - the directive being that you can't call setArguments after the fragment is set up; the solutions to both cases are similar**)
3)hypothesis: is the problem that setArguments is called twice?
4) the participant adds code to getArguments before setting the arguments and comments out the setArgument call

5) the participant tests the application and find that the application still crashes

7) the participant sees the null pointer error and determines that the fragment may already be active at that time

3) **hypothesis: is the problem that the fragment is already active?**

3) how do I know if the fragment is already active?

7) the participant Googles “android how do I know if a fragment is already active”

7) the participant reads through some questions and answers

7) the participant finds a way to check if the fragment is visible

4) the participant implement code to check if the HeadsUpFragment is visible before creating the Bundle and only updates the Bundle if the fragment is visible

7) the participant tests the application and finds that the application crashes with a “Fragment is already active error message”

2) the participant concludes that isVisible didn’t test the right thing

4) the participant guesses that isAdded should be the check instead

7) the participant tests the application and finds that the application dies with a nullPointerReference

7) the participant goes back to reading old tabs the participant left open for other ideas

2) the participant wonders if the question answer is really the correct answer since the answer was never accepted by the question asker

2) the participant comments that the participant could use print statements in different locations but decides that is not the best way to go

4) the participant changes the check condition to check if the arguments are null. If the arguments are null, then the Bundle is created, otherwise the participant just adds to the Bundle (1:25:00)

7) the participant tests the application and finds that the application crashes with a Fragment already active error

7) the participant debugs the application and finds that the code goes through the arguments are null branch

2) the participant comments that this does not be the right way to solve the problem

4) the participant changes the getArguments == null to isInLayout but stops before finishing

7) the participant Googles the Fragment APIs for examples of setArguments and getArguments and reads through them, then checks that is the code is fine

7) the participant Google “how to know if a fragment is already active”, sees that the answers are similar to what he searched before, and then goes back to the code

4) the participant changes the isVisible to not isAdded and states that this attempt is just a guess

7) the participant tests the application and finds that it still crashes

4) the participant changes not isAdded to not isVisible

7) the participant tests the application again and get the same crash (1:37:10)
7) the participant searches for other autocomplete options to find if the fragment is already active
7) the participant goes back to reading a previous stack overflow answer
7) the participant reads through the setArguments documentation in an answer
7) the participant sees an answer to setCustomArguments(Bundle)
3) is the problem that setCustomArguments bundle is not overwritten?
4) the participant tries to override setCustomArugments and set the arguments using this method
2) the participant quits halfway through and decides that the problem is taking too long to solve
2) the participant comments that it will probably take about 20-25 minutes more to fix the issue

Participant 4, Task 2, Fault 29
7) the participant reads through the application to understand the prompt
7) the researcher demonstrates the problem in the application
7) the participant asks clarification questions about the problem and the test cases (21:50)
7) the participant sees the warning message in HeadsUpNotificationFragment: **This fragment should provide a default constructor (a public constructor with no arguments)**
(com.example.android.lnotifications.HeadsUpsNotificationFragmment)

From the Fragment documentation:
Every fragment must have an empty constructor, so it can be instantiated when storing it's activity's state. It is strongly recommended that subclasses do not have other constructors with parameters, since these constructors will not be called when the fragment is re-instantiated; instead, arguments can be supplied by the caller with setArguments and later retrieved by the Fragment with getArguments().
2) the participant reads the message and concludes that the application needs an empty constructor
3) hypothesis: can the application be fixed by creating an empty constructor?
4) the participant comments out the constructor with an argument and creates an empty constructor
5) the application notices that the time isn't set in the application so that problem aso has to be fixed
3) can the time be set with a setMethod?
4) the application creates a setMethod for the fragment and then passes the start time to it
7) the application tests the application and figures out that it works (28:00)

Participant 4, Task 1, Fault 31
7) the participant reads through the prompt to understand the problem
7) the researcher starts the application but lets the participant step through the problem
3) the participants asks what exactly is a set meeting message?
7) the researcher tells the participant to step through the problem process
7) the participant goes to MeetingFragment
2) the participant guesses that there is something in the onCreate method
7) the participant finds the button's onClick method
3) is this button the set meeting button? (33:16)
7) the participant comments that the display string should be modified before it is displayed
3) what is makeText?
    7) the participant right clicks the method to go to the method declaration and determines that it is provided Android method
2) the participant comments that this task is much harder than the previous one
7) the participant looks through the code for the startTime button and the endTime button
7) the participant looks though the TimeSelectionFragment
2) the participant comments that the participant isn't sure when the value is stored
4) the participant starts to add getters and setters for the time selection fragment but the researcher comments that those variables aren't supposed to be there (they are never set)
7) the participant reads through the TimeSelectionFragment and then reads through the TimePickerFragment after finding a reference to that fragment in the TimeSelectionFragment
2) the participant determines that the TimePicker fragment should pass the data to the TimeSelectionFragment and then to the MeetingFragment
3) should the data be passed between the three fragments?
    7) the participant notices that the getters and setters in the TimePickerFragment are not used
    4) the participant starts to change the onClick handling in the TimeSelectionFragment to get the selected hour and selected minute
    4) the participant then adds getters to the new times in the TimeSelectionFragment
    4) the participant adds meeting start times and meeting end times
    4) the participant sets the meeting start and end times to the button values, which means that the second value overrides the first
    2) the participant then notices that the start and end time would be the same
    4) the participant removes the end time line (48:05)
    4) the participant changes the initial displayed button text to the start time and end time
    3) how can I pass a parameter to the onClick method?

4) the participant adds the variable that the participant would like to pass to the onClick method
7) the participant reads the compile error that the variable could not be initialized at this point
4) the participant adds default initialization value of the string as a blank string
7) the participant finds that the error is still there
4) the participant deletes the variable reference
7) the participant goes back to reading through the file to try to figure out how to pass the information
4) the participant changes the variable types from private to public static and access the variables with a static reference
7) the participant tests the application and finds that the meetingStartTime and meetingEndTime are being set to 0 strings (54:29)
7) the participant looks at how the buttons are added and decides to look up “Android fragment commit()”
7) the participant reads the method’s description online
4) the participant tries to print out the selected times at onClicks but has difficulty finding the information in all the other android print statements so the participant deletes the line
4) the participant then tries to print out the time onTimeSet
4) the participant finds that the times are being printed correctly
4) the participant adds code to get the selected time from the TimeSelectionFragment
4) the participant tries to set the final button times in the onViewCreated method, where the buttons are clicked.
7) the participant tests the application and the application prints 0’s
2) the participant comments that the participant isn’t sure about the order that methods are called
8) the researcher calls the task
2) the participant predicts that the task will take 10 to 15 minutes (1:08:30)
8) the researcher allows the participant to resume the task based on the short estimated time
7) the participant looks through the time selection fragment and finds that the descriptions are different for the different fragments
4) the participant adds code to print out the description of the fragment arguments
7) the participant finds that the descriptions are different for the different fragments

4) the participant sets the meeting starts and the meeting end time using the description tag to set the right one in onTimeSet in the TimePickerFragment
7) the participant tests the application and finds that the time is correctly printed (1:15:00)

Participant 4, Task 4, Fault 13
8) the participant reads through the prompt to understand the problem
7) the participant executes the application to see the problem
7) the participant reads through the stack trace and decides to look into the OtherMetadataFragment, which is mentioned at the top of the stack trace
7) the participant scrolls through the OtherMetadataFragment
7) the participant tries to look up the method declaration of the failing getActivity call and determines that the method is from the Android Fragment class so it is unlikely to be incorrectly implemented
7) the participant looks up the Fragment's getActivity method using the documentation pages the participant was already on
3) hypothesis: is the problem that the fragment is incorrectly associated with an activity? (the participant sees that getActivity returns the associated activity from the documentation)
7) the participant then tries to look up the method definition for setTabListener
7) the participant then looks through the Fragment documentation to see how to associate the Activity
3) does the application work if the getActivity call is removed from the displayed string?
7) the participant finds the activity title is set in the onCreate method, so the participant considers copying the title string to the displayed message
4) the participant replaces the displayed message
5) the participant tests out the application and finds that the application still crashes
4) the participant removes the changes
7) the participant tries to look at all the Fragment class methods
4) the participant looks through the different lifecycle methods of the activity and finds the location that displayActivityTitle is called
7) the participant tries to understand when the onTabSelected method is called
3) hypothesis: is the Activity not created yet when displayActivityTitle is called?
4) the participant comments out the call to displayActivityTitle
7) the participant tests the application and determines that the application now works
7) the participant decides to wrap the displayAcitivityTitle in a try catch block to print out the exception without causing the application to crash
3) hypothesis: can the title print method call be moved to the Activity?

4) the participant moves the message to the activity
8) the researcher calls the experiment due to time
2) the participant comments that it probably would have taken another ten minutes to solve the problem.

7,4,1
Participant 5, Task 7, Fault 12
7) the participant reads through the problem to understand the problem
7) the researcher runs the program to demonstrate the crash
2) the participant comments that the stack trace seems to imply that that application is crashing before the the user written code is reached
2) the participant finds that the process is named after the activity in the user defined section of the application and finds that the activity is mentioned in the Runtime Exception error message
3) hypothesis: the application is compiling?
8) the participant asks the researcher to confirm the application compiled because the participant is not very familiar with AndroidStudio
8) the researcher confirms to the participant that the application compiled
7) the participant runs the application in debug mode to try to find the failing line
8) the participant expresses disappointment in AndroidStudio automatically catching the RuntimeException and starting the debugger like Eclipse does by default
7) the participant reads through the stack trace, looking for clues
7) the participant opens the Activity and quickly glances through the initial set up code
7) the participant goes back to the stack trace.
7) the participant runs the application again
7) the participant searches through the project for “Unfortunately”
7) the participant concludes that the error message must be an Android default message, since it doesn’t seem to be in the project
2) the participant reads that you must call removeView first but doesn’t believe that is necessarily the way to fix the problem in this case.
7) the participant rereads the prompt to learn if it is possible to copy working code from the other fragments
7) the participant runs that application again
2) the participant comments that the stack trace from running the application again is unhelpful.
7) the participant tries to run the application by right clicking on the application and gets the same error message
7) the participant Googles the “Caused by java.lang.IllegalStateException: The specified child already has a parent. You must call removeView() on the child’s parent first.” line.
7) the participant clicks the first link which is a stackoverflow answer. **The first answer state that the inflate method should use the false parameter.**
3) hypothesis: is the problem that that inflate method doesn’t have the false parameter.

4) the participant adds the false parameter
7) the participant tests the application and finds that it works
2) the participant mentions that the participant had no idea why that worked

Participant 5, Task 4, Fault 13
7) the participant reads the prompt to understand the problem
7) the researcher demonstrates the crash
7) the participant starts to read through the stack trace
7) the participant tries to go to the declaration of getActivity.
7) the participant tries to search for setActivity but doesn't find anything
3) where is the failure?
7) the participant then realizes that the participant had been searching through the Activity, which the researcher opened, and changed to the OtherMetadataFragment, the top class mentioned in the stack trace
7) the participant searches for activity in the OtherMetadataFragment
3) was the application working before trying to add the feature?
7) the researcher confirms that the prompt is implying that the application worked before trying to add the new pop up on the Others tab
2) *this seems to lead the participant to believe that the other getActivity calls work*
3) hypothesis: does the application work if the makeText line fails
4) the participant comments out the makeText line
7) the participant runs the application and finds that it works
3) hypothesis: would the application work if the getTitle call was hard coded?
7) the participant looks through the fragment for the title
2) the participant asks if that solution is acceptable
8) the researcher answers that it is not the prefered method
8) the participant decides to abandon this route
7) the participant looks into the getActivity call and finds that the method stub only seems to throw an exception
2) the participant seems confused
7) the researcher comments that the method doesn't seem right and the participant should look it up online
2) the participant comments that the method isn't the problem, because it works in other locations in the code
7) the participant sets a break point in the display activity title
7) the participant executes the code to the break point
3) why would getActivity work else where and not here
7) the participant changes to LNotificationActivity
2) "oh, because this is the first time it being used" referring to the getActivity call in displayActivityTitle being the first time that getActivity is called in the OtherMetadataFragment
7) the participant searches for other uses of the displayActivityTitle method and doesn't find anything

2) the participant concludes that that the method is being executed too early
3) when is the fragment attached to the activity?
7) the participant concludes that the tab is created correctly after looking at earlier code
3) when else is getActivity called?
    7) the participant finds that getActivity is used in onViewCreated in the OtherMetadataFragment
3) what is calling the different lifecycle method in the onViewCreated?
3) hypothesis: would the getActivity call work if the fragment has already been instantiated?
    4) the participant adds code to skip the Toast call the first time but execute the second time
    7) the application crashes when opening the tab the second time
    5) the participant concludes that call does not work if the fragment is instantiated before.
7) the participant looks into the getActivity calls in other Fragments
7) the participant scrolls through the Activity looking for other ideas
3) what is fragment in this context - referring to the onTabSelected method
2) the participant figure out what fragment is - a reference to the new tab
3) should the makeText call be moved to a different location
    4) the participant moves the call to the showNotificationClicked method
    7) the participant finds that the title shows when the button is pressed
    3) that is the title? - the participant originally thought the fragment title should appear
    2) the participant realizes that the activity title is the application title, not the fragment title
    3) why does the getActivity call work when it is moved to the button?
3) hypothesis: by that point activity has been set right?
    4) the participant changes the code to display the popup the second time the tab is opened
    7) the participant steps through the application, first clicking the popup and getting the title, then changing off the tab and back
    5) the application still crashes
3) how is that possible? - that getActivity will fail when it has worked before
7) the participant looks through OtherMetadataFragment for ideas
7) the participant looks for usages of showNotficationsClicked
7) the participant Googles “getActivity returns null”
7) the answer states that the activity should be saved in onAttach
3) hypothesis: does the OtherMetatadata fragment not execute onAttach when it should?
    4) the participant implements an onAttach method that prints a statement
    7) the participant puts a break point in the onAttach method
    7) the participant runs the application in debug mode
    7) the participant realizes that the participant forgot to override the super method

4) the participant implements the super method call
7) the execution reaches the onAttach method debug point first because the displayActivityTitle debug point is skipped the first time
7) the participant continues the application until the execution crashes
5) the participant concludes that his theory is incorrect

7) the participant goes back to the stack overflow answer
8) the participant comments that the answer is doing what the participant didn't want to do
2) the participant comments that the participant doesn't think that method will work but decides to try it anyways
3) is the problem that the activity should be saved in onAttach?

4) the participant changes the code to save the Activity in onAttach and removes the code to only execute the Toast message the second time
7) the participant then executes the application
5) the application still crashes (1:05:28)

7) the participant goes back to the stack oveflow question
7) the participant then looks up find usages of onAttach but doesn't find any in the project
2) the participant comments that the participant thought the onAttach saving would work for sure when executing the makeText call the second time
7) the participant realizes that the Toast message executes the first time
3) hypothesis: will the application work if the tab is opened first and the activity is saved in onAttach

4) the participant changes the code to make the toast execute on the second occurrence of opening the tab
7) the participant tests the application
8) the participant comments that this error is very confusing
7) the participant finds that the message appears in this case

7) the participant realizes that the FragmentTabListener is an inner class
3) can the activity be passed by calling onAttach before the framework calls it?

4) the participant adds code to call onAttach when creating the FragmentTabListener
7) the participant tests the application and determines that the application still works
4) the participant removes the code to only execute the message after the first time
7) the participant tests the application and determines that the application crashes
4) the participant changes the code to specifically call the onAttach method if the fragment is the OtherMetadataFragment
7) the participant tests the application and finds that it fails
2) the participant concludes that the saved activity must be cleared at some point

8) the researcher asks if the participant wants to end the task
8) the participant decides to quit
2) the participant comments that the solution would take different amount of time depending on if the participant needs to understand the effects of the change or not
2) the participant estimates that the task would take another hour to finish

Participant 5, Task 1, Fault 31
7) the participant reads through the prompt to understand the problem
7) the researcher demonstrates the application
7) the participant searches for “default value” in the project
7) the participant looks at the location where default value is hard coded in the code
3) how can the participant get the time where it needs to be?
8) the participant gets confused about the arrow icon in the code, then clicks it and realizes that it is masking an inner class definition; the participant comments that this feature makes the code more difficult to understand
7) the participant reads through the timePickerFragment to see how the time is set and finds the get time methods in the class
7) the participant tries to understand the onClick listener
2) the participant comments that the TimePickerFragment is oddly not part of the meeting fragment
7) the participant tries to find instances of when onTimeSet is called but finds that there are none in the project
3) is onTimeSet dead code?
4) the participant adds code to the method that would generate a runtime exception
7) the participant tests the application and finds that that the app crashes
5) the participant realizes that the method is called by the framework and the participant had only searched project files earlier
7) the participant looks through the TimeSelectionFragment
3) does the meeting fragment have a reference to the TimeSelectionFragment and does the TimeSelectionFragment have a reference to the TimePickerFragment?
7) the participant finds that that the MeetingFragment uses TimeSelectionFragment
3) how could the participant get the data from the TimeSelectionFragment if the TimeSelectionFragment has the time?
7) the participant looks for the meeting fragment in the TimeSelectionFragment
3) can the TimeSelectionFragment be referenced in the onClickListener?
5) the participant tries to add code to reference the TimeSelectionFragment in the onClick listener but finds that that reference isn’t found by the compiler so the participant stops
7) the participant looks through possible methods for the TimeSelectionFragment but doesn’t find a call that the participant likes

2) the participant comments that the solution seems to involve writing a lot of code for a study
2) the participant determines that the participant has to change the value when onTimeSet is called
3) hypothesis: can the values be stored in the meeting fragment and set in the TimePickerFragment?
3) which button is which?
8) the participant comments again that the participant wouldn't do this at work
2) the participant determines that the code covers multiple buttons
2) the participant determines that that MeetingFragment has two TimeSelectionFragments
3) why does the code look like there are is only one button?
7) the participant notices that both buttons are initialized in a similar method
3) hypothesis: should the button instances be separated?
4) the participant separates out the button initialization and saves them to an array
4) the participant tries to reference the buttons in the onClick method but finds that they are not found in the context
7) AndroidStudio displays a warning that the array must be declared final to be referenced in the inner class
4) the participant changes the TimeSelectFragment array to final
7) the participant tries to get the Button text but finds that the method is not implemented (1:35:50
3) is there a reason that getButtonText is implemented for the button
4) the participant creates a public variable for the buttonText
4) the participant sets the button text in the TimePickerFragment
4) the participant adds code to extract the button text
7) the participants the application and finds that it works

Participant 11, Task 1, Fault 31
8) the participant reads through the prompt to understand the problem
7) the participant runs the application to see the problem
3) where is are the meeting values being set? - the participant comments that the participant plans to debug how the values are set
7) the participant searches for where the time is set in the MeetingFragment
7) the participant gets confused about where the process is starting
7) the participant determines that the confirm meeting button is the set meeting button
4) the participant adds a comment that the confirm meeting button is the set meeting button
7) the participant looks for uses of the button and finds that the button is using a default print message
2) the participant comments that this onClickListener is the problematic area

2) the participant comments that the participant decided to not use the debugger like the participant planned because the participant found the problematic area and can understand the code by looking at it
3) can the participant confirm this method is the method which prints the message?
    4) the participant changes the print results
    7) the participant runs the application and finds that the print works
7) the participant tries to find the two select time buttons, and looks in the TimeSelectionFragment
7) the participant determines that the onClickListener for the TimeSelectionFragment is creating a TimePickerFragment
2) the participant incorrectly assumes that a line is setting the time
7) the participant investigates the line and finds that the line didn't do what the participant thought
7) the participant sets a breakpoint in the method to try to find where the time selected is but before testing it, the participant determines that that the breakpoint isn't needed there
2) the participant deduces that the TimePickerFragment is selecting the time by looking at the TimeSelectionFragment and the TimeSelectionFragment is only starting the TimePickerFragment (34:16)
7) the participant then decides to debug the method that the participant was trying to debug earlier, so the participant can determine what variables are set in that method so to the participant can get an idea about how to get the time
2) the participant becomes confused about the framework control flow while debugging; the participant wonders why the debugger when into the Android OS, and didn't realize that the framework was calling the method and that is what the participant stepped into
7) the participant looks into TimePickerFragment and finds getSelectedHour, getSelectedMinute
2) the participant concludes that those methods are the one the participant needs to use
3) how can the participant get a reference to the TimePickerFragment?
3) how can the participant link the TimePickerFragment to the MeetingFragment?
3) hypothesis: does the TimePickerReference in the onClick method in TimeSelectionFragment contain a non-null instance?
    7) the participant tries to debug the method the participant debugged again to determine if the reference to the TimePickerFragment could be found in that method
    2) the participant comments that the participant wants to insure a value is there so not to waste the participant's time
    7) the participant looks up the declaration of getFragmentManager in hopes to get the TimePickerFragment
    7) the participant steps through the method and determines that the TimePickerFragment is instantiated correctly

2) the participant also learns that the method finishes before the participant is able to set the time

2) the participant commented that the participant learned that this method isn't the method the participant wants to change

7) the participant investigates the TimePickerFragment in hopes to find if the time is set

3) hypothesis: is the TimePickerFragment setting the selected time?

4) the participant adds a print statement to show the selected time

7) the participant runs the application

2) the participant comments that the debugging is too slow to check for something this small

7) the participant finds that the time is set in that case

2) the participant comments that the TimePickerFragment is working correctly and the participant now just needs to get the selectedHour and the selectedMinute

3) how can the participant set something in MeetingFragment?

7) the participant looks through the MeetingFragment for possibilities

7) the participant tries to find if anyone is using the TimePickerFragment, specifically the getSelectedHour and getSelectedMinute (50:00)

2) the participant comments that the participant is either trying to figure out how to access the MeetingFragment to save the times, or called the saved times in the TimePickerFragment from the MeetingFragment

2) the participant concludes that the MeetingFragment must access the TimeSelectionFragment in the button, which means the participant can pipe the classes together to get the information

3) hypothesis: can the selected times can be piped through the reference for the buttons?

4) the participant adds local time variables to the TimeSelectionFragment class

4) the participant tries to set the time in the onClick method in the TimeSelectionFragment

2) the participant comments that if the method is called twice, then the participant is in trouble.

4) the participant adds print state for the starthour and startminute variable the participant added

7) the participant has difficulty with the inner class setting the starthour and the startminute and currently gets a compiler error

2) the participant is confused about why the compiler error for the inability to access the variables in the inner class occurred because the participant doesn't realize the button code is inside the button inner class. (57:25)

2) the participant determines that the button is inside the inner class

3) how do I save the state? - the participant is asking how to reference the outside state in the inner class

2) the participant initially assumes that the participant could used the TimePickerFragment to access the state of the meeting fragment - I am unsure why
4) the participant adds code to the TimePickerFragment to get the TimeSelectionFragment
4) the participant tries to print the values of the TimeSelectionFragment in the onClick method of the TimeSelectionFragment by getting the reference from the TimePickerFragment but then stops - the participant realizes that the problem was the references were still private
4) the participant changes the variables references to public
4) the participant accesses the variables in the button and sets them to the values from the TimePickerFragment.
2) the participant realizes the same method call handles both buttons, so the participant is trying to figure out how to differentiate the two buttons
4) the participant adds code to print the values
7) the participant runs the application and finds that the values print 0's at that time and both print when clicking on the different buttons
2) the participant concludes that the buttons are not setting the values returned
4) the participant replaces the set variables with the get calls for the TimePickerFragment
7) the participant runs the application again and finds that the value is 0 again
2) the participant concludes that the TimePickerFragment is not setting the values correctly or the TimePickerFragment references the participant has is not the TimePickerFragment the participant wants
2) the participant runs through what the participant knows which is that the timeSelectionFragment contains the buttons, each button is TimePickerFragments, the TimePickerFragments set the time correctly, but the TimePickerFragment the participant references in onClick is not the correct reference that the participant expects
7) the participant looks into the beginTransaction method
2) the participant comments that the method does something different than the participant expected. The participant comments that the participant hoped the FragmentTransaction set the time values, but the values seem to be set later
2) the participant comments that the participant needs to find when the time values are set
7) the participant looks for a show method in the TimePickerFragment
2) the participant decides to move the piping to the end of when the values are set in the TimePickerFragment
4) the participant moves the code to the end of the TimePickerFragment method which sets the time
2) the researcher learns that the participant is colorblind, which may have been the cause of the compiler errors
7) the participant tests the print statements and finds that they are working

2) the participant thinks the next step is to get a reference to the TimeSelectionFragment in the MeetingFragment
4) the participant adds local variables to the meetingFragment to hold the time
4) the participant decides to add a check to the addButtonToView method to see which button is calling the method
4) the participant tries to save a reference to the different buttons in the method using the button text to differentiate the buttons. The participant saves each button as a class variable. (1:26:22)
7) the participant tries to reference the object in the button handling but gets a compiler error and decides to erase the code.
7) the participant tries to see if the participant can reference the MeetingFragment statically or if the MeetingFragment can be referenced from the FragmentManager
3) how can the participant pass the information into the onClick method?
7) the participant looks into the Button class for ideas about how to get the time information into the buttons
7) the participant looks into the fragment_meeting.xml but quickly quits when the participant sees that the design is starting to display
7) the participant search online for “android pass extra argumnets to button onclick” and clicks on the first stackOverflow answer
7) the participant finds an answer that wraps the context of the button inside another method and sets the parameters of the method to final.
3) can the participant use the wrapping method technique to reference the variables in the onClick method?

> 4) the participant copies the answer to the question and then adjusts the answer to the current variables in the Fragment
> 7) the participant tests the application and finds that the application works except that the participant was using dummy text
> 4) the participant changes the application to remove the dummy text and replace it with the time references
> 7) the participant tests the application and finds that it returns 0s
> 4) the participant changes the arguments to pass in a TimeSelectionFragment
> 4) the participant also changed the parameter to final
> 7) the participant tests the application and finds that some values appear but they seem to be incorrectly formatted
> 4) the participant realizes that the participant was treating the hour and minute as separate times, so the participant adjusted the code to put the hour and minutes into the same print out
> 7) the participant tests the application and finds that the start time printed but the end time didn’t
> 2) the participant realizes that the end time was not correctly referenced,

4) so the participant changes the method to take both TimeSelectionButtons in the context and changes the code to work with the the new application.
7) the participant tests the application and finds that the application prints out the time correctly

Participant 12, Task 2, Fault 29
7) the participant reads through the prompt to understand the problem
7) the participant runs the application to see how it works and where the time appears
7) the participant reads the error message and sees that there is no default constructor
2) the participant states that the participant is trying to understand how the HeadsUpNotificationFragment uses the appStartTime
7) the participant seems to give up and goes back to reading the error message
3) is the application current executing the HeadsUpNotificationFragment?
4) the participant adds a print out to the current constructor
7) the participant tries to look for where the HeadsUpNotificiationFragment constructor is called
7) the participant finds that the constructor is called in the newInstance method
2) the participant realizes that the constructor and the class have the same, *so the participant seems to get confused and thinks that the constructor with a parameter is a default constructor*
2) the participant comments that the participant realizes that the constructor is called
3) hypothesis: would creating a default constructor fix the problem?
4) the participant creates a default constructor
7) the participant runs the application to see the results
5) the participant comments that the participant hoped the red lines would go away
7) the participant also checked the print output for the constructor print message and found that the print message appeared.
7) the participant reads the error message more in depth: **Avoid non-default constructors in fragments; use a default constructor plus FragmentArguments(Bundle) instead**

From the Fragment documentation:
Every fragment must have an empty constructor, so it can be instantiated when restoring it's activity's state. It is strongly recommended that subclasses do not have other constructors with parameters, since these constructors will not be called when the fragment is re-instantiated; instead, arguments can be supplied by the caller with setArguments and later retrieved by the Fragment with getArguments().

2) the participant decides to create a getter and setter function instead of using a Bundle - the participant later comments that the participant is more comfortable with that solution, so the participant wants to try that reference first
2) the participant is worried that the constructor call may be referenced at various points in the code, and changing it without understanding those other cases could cause issues.
3) what would be the repercussions of changing the constructor?

7) the participant uses the AndroidStudio usages and finds that there is only one usage

4) the participant gets confused for a bit and starts trying to create a constructor with a different name, which isn't allowed in Java

4) the participant changes the constructor back to a single default constructor after noticing the problem

4) the participant creates a getapptime method to set the app time for the class instance

4) the participant tries changing the original constructor to a method but then has a problem with creating the new object so the manually undoes the changes

4) the participant sets the object's time using the getapptime method in theInstance method of HeadsUpNotification (48:08)

4) the participant replaces the original constructor call with the new constructor call

7) the participant tests the application and finds that it works

Participant 12, Task 4, Fault 13
7) the participant reads through the prompt to understand the problem
7) the participant runs the application to see the crash and read through the stack trace
7) the participant reads that the problem is a null pointer exception and assumes the title is null
3) is the title not being set in the application?

7) the participant looks into the Activity class to see if the title can be set to null or return null

4) the participant creates a print statement in the displayActivityTitle method to make sure that is the location of the problem

7) the participant is trying to see if the participant can set the title and the participant is confused about where to do that in the control flow of the application

4) the participant decides to replace the getActivity().getTitle call with a reference to variable

2) the participant gets confused with the Object type required to replace the getTitle() call.

4) the participant looks at the makeText method and sees that it takes a CharSequence. The participant tries to create a charSequence object but finds

that the compiler throws an error (because the participant is trying to instantiate an interface)
7) the participant decides to google “charSequence declaration” to create an object of type CharSequence, the type that the makeText method takes as a parameter. (1:01:48)
7) the participant clicks on the first StackOverflow question to figure out where the CharSquence class is defined, the answer says the class is from the java.lang package
4) the participant decides to revert the use of the title variable and replaced it with the call to getActivity().getTitle().
7) the participant runs the application and looks through the stack trace to find the print statement in that method
2) the participant concludes that the makeText line is the failing method and the participant is changing the correct location
7) the participant decides to read into the Toast class (1:04:38)
4) the participant tries to import CharSequence into the code
7) the participant then tries to look for an example in Activity.java and finds to java.lang imports
2) the participant concludes that one of those imports are the correct way to import the CharSequence type
4) the participant tries to import the CharSequence type using those import statements
7) those imports didn’t work so the participant went back to looking through the Activity.java file
4) the participant decides to remove the getActivity((().getTitle() call from the line and finds that the line compiles.
7) the participant then tests the application and finds that the application still crashes (1:12:06)
2) the participant concludes that that application requires a CharSequence and can’t use the String, which caused the run time exception (without looking into the exception in depth)
3) does the program actually call makeText or is the null value thrown before.
    4) the participant adds print code to the makeText file inside the Toast class
    7) the participant runs the application and doesn’t find the makeText print statement
7) the participant Googles “charSequence declaration in Java” and clicks on the first StackOverflow question about converting a String to a CharSequence, which says that a String is a CharSequnce
4) the participant tries to add if null handling for the title (1:17:45)
2) the participant comments that the participant still believes the title is not being set.

2) the participant gets confused about the compiler error message but then realizes that the participant should handle the else case after rereading the error case

4) the participant implements code to set the title to an empty string if the title is null

7) the participant runs the application and finds that the application still crashes

3) hypothesis: is getActivity returning null?

4) the participant decides to add a print statement to test if the getActivity call is returning null

4) the participant decides to only test the title if getActivity does not return null but the participant gets a compiler error because title is not initialized in all paths

4) the participant fixes the problem by initializing title at the start menu

7) the application still crashes

5) the participant finds that the Activity from getActivity is null

7) the participant reads that the getResources() call is failing in the error message, so the participant tries to figure out what that means

2) the participant finds that the other getActivity call in makeText is returning null

2) the participant comments that the participant believes that getActivity should not be null at this point

7) the participant tries to look through the OtherMetadataFragment for where the title is set incorrectly.

7) the participant looks through ContextThemeWrapper since that is the failing line (1:41:44)

7) the participant looks back through the stack trace for ideas

3) is getActivity always null before the application crashes?

4) the participant tries to print out the value of getActivity in the first location the getActivity call seems to occur for the participant, the onCreate method

2) the participant isn't sure if getActivity should return a non-null value at this point

7) the participant runs the application in debug mode but the application crashes like normal

7) the participant doesn't find the print result and so the participant

4) the participant deletes the getActivity line

7) the participant looks into how the displayActivityTitle is called

2) the participant believes that the flow of the application is incorrect, because the getActivity call is not set when it is used, but the Activity seems to be set by the Android Framework - **getActivity will return null if called before the Activity is set**

7) the participant runs the application again and sees the crash

4) the participant adds a print statement before the displayActivityTitle call

3) why does displayActivityTitle only fail for the others tab?

7) the participant sets a debug breakpoint the onTabSelected method

7) the participant runs the application in normal mode and finds that the prints statement in the method executes
7) the participant prints out the value of the activity for the fragment when the tabs are about to change, the participant finds that getActivity returns null
3) what does getActivity return in onTabSelected for the other tabs?

- 7) the participant tries to find what getActivity returns when the other tabs are selected?
- 7) the participant tests the print statement but forgets that the print statement is within a check that only prints for the OtherMetadataFragment
- 7) the participant looks for uses of onTabSelected in the project but doesn't find any
- 2) the participant is still confused why the method didn't execute for the other tabs and is looking for ideas how to fix the non printing problem
- 7) the participant looks into the code for the other tabs for uses of Activity to see if the Activity handling is any different
- 2) the participant figures out the problem - that the method is only called for the other metadata fragment

4) the participant comments out the displayActivityTitle call
2) the participant comments that the participant doesn't understand why the call is only for the others tab
7) the participant tests the application and finds that it works
2) the participant asks the researcher what the title is in the application, which the researcher tells the participant.
2) the participant wants to hard code the output to remove a getActivity call
2) the participant comments that the participant forgot where the activity title is set
7) the participant tries to search the project for the Activity title but doesn't find it after looking through a few files and searching a couple of files with control f
3) hypothesis: mActivity is not public so that is the problem?

- 4) the participant changes mActivity to public in the test file (the participant becomes confused with the main application file and it's test
- 7) the participant tests the application and finds that the application still crashes with the same error message

7) the participant searches for getActivity
8) the participant decides to stop the study
8) the participant estimates that the code would take 2 more hours to fix because of the participants confusion with the new IDE

Participant 13, Task 3, Fault 17
7) the participant reads through the prompt to understand the problem
7) the participant runs the application to see the options menu not appearing
3) hypothesis: Can the options menu be added with the design tab for the xml in AndroidStudio?

7) the participant navigates to the acitivity_notification.xml

3) how is the layout structured?

7) the participant tries to scroll through the default widget options but doesn't see it

3) the participant asks how to change from the design tab to the text tab

7) the participant changes to the text tab to read the xml

7) the participant goes back to looking through the default widget options

5) the participant notices that there isn't a menu item

2) the participant remarks discovering that there is a difference between the activity and the fragment for how to handle things like the options menu

7) the participant searches online for "clickable menu android"

7) the participant sees an answer about how to add menu items to the options menu, but it doesn't seem to tell the participant how to set up the options menu.

7) the participant decides to Google "how do I add a clickable menu in fragment of android" and selects the first stackoverflow answer

7) the participant reads through some of the answers - one of the answers says that a **you must call setHasOptionsMenu(true) in onCreate** but the participant only glances at it and then moves on without realizing it is the answer. Instead the participant assumes that the answer is in the question.

2) the participant assumes that the to code to create the menu is not in the project.

2) the participant would prefer to use the drag and drop editor compared to taking the code out of question

7) the participant tries to look through the widget options again in the design tab but still doesn't see the menu editor (21:44)

7) the participant Googles "clickable menu design in android using design in xml"

7) the participant clicks on the official Android menu documentation and scrolls through the page.

47 the participant stops to read about how to create an option menu for an Activity.

3) hypothesis: is the problem that the menu is not defined in the xml file?

4) the participant finds an example about how to define the menu items in the xml and tries to copy that to the project

2) the participant has difficulty determining where to define the menu in the xml

7) the participant checks the drag and drop options again to see if there is an easier option.

4) the participant decides to paste the sample menu in the acitivity_nofication.xml file

5) the added code throws a compile error so the participant deletes the pasted code

7) the participant goes back to reading the documentation to figure out where to add the menu

4) the participant finds another example of defining the xml in the documentation and copies the example

4) the participant creates a new menu file and pastes the online code into the new xml file but the drawables don't compile

2) the participant assumes that the problem is that the new menu is not correctly connected to the rest of the application

2) the participant describes that the participant understands that the R directory defines all the elements and then each element can be accessed through the element's id in the xml file and so the participant is now trying to extend a listener to add to the menu to the application. The participant changes back to the documentation and explains his understanding of how to the example handles the different button clicks.

3) the participant explains that the participant is trying to understand how the application is set up and where you can add the menu to the application

3) the participant explains that the participant is unsure how to put the options menu in the correct spot in the layout.

2) the participant looks through the xml text to see how the elements are laid out in the layout and to see how the information for the different items are set

4) while explaining what the participant is doing, the participant tries to add a button where the options menu should be, but finds that he can't put the button there. The participant then deletes the incorrectly added button.

2) the participant comments that the developer guide says to add a menu with the onCreateOptionsMenu but the participant isn't sure that that menu will be added in the right spot

3) hypothesis: adding onCreateOptionsMenu will automatically put the options menu in the correct spot.

8) the participant asks for water and after a discussion, the researcher leaves to get the participant water

7) the participant reads through the Activity file to figure out where to add the onCreateOptionsMenu

4) the participant decides to add the onCreateMenu to the FragmentTabListener class but finds that the method doesn't compile correctly

4) the participant decides to move the method so the original method paste was deleted

8) the researcher comes back with water

7) the participant goes back to the developer page and reads about how to use the onCreateOptionsMenu call

7) the participant briefly looks into the Activity class docs but then goes back to reading the developer's guide

2) the participant defines that the participant needs to override onCreateOptionsMenu in the Activity

2) the participant also explains that the participant is still confused about if this approach will work because it doesn't define the options menu in the xml, but wants to try what is in the developer's guide

7) the participant searches for “example of oncreateoptionsmenu used in activity android” and clicks on the second link because the first link is the developer’s guide that participant was on earlier.

7) the participant reads the answer and determines that the onCreateOptionsMenu goes in the main Activity class

4) the participant pastes the onCreateOptionsMenu method in the Activity and finds that the menu xml reference is broken

4) the participant changes the menu name to user defined menu

7) the participant tries to run the application but finds out the user defined menu is not compiling because the drawable references are broken

2) the participant determines that the problem is the broken drawables

3) Where can the drawables be found in the project? - the participant finds the answer

4) the participant changes the icons to drawables in the project

7) the participant tests the application and finds that the images appear in the correct place - however the options menu was added in the Activity and not the fragment.

7) the researcher decides that the participant has solved the task

2) the participant comments that the participant had experience with this a while ago but has forgotten some of it.

Participant 14, Task 3, Fault 17

7) the participant reads through the prompt to understand the problem

7) the participant runs the application to see the crash and read through the stack trace

7) the participant reads through the main Activity

7) the participant then reads through the HeadsUpNotificationFragment

7) the participant finds the onCreateOptionsMenu

7) the researcher mentions that the HeadsUpNotificationFragment functions are called by the Activity

7) the participant Google searches for “android 3 clickable menu”

7) the participant looks though the menu developer guide

2) the participant get a new idea that may be more profitable to investigate

3) hypothesis: can the options menu can be added with the gui interface?

7) the participant has trouble finding the gui interface

3) the participant asks the researcher about how to access the gui interface and so the researcher tells the participant

4) the participant tries to add a button but it is not the right button and not in the correct space

5) the participant realises that the button can’t be put in the correct position with the editor

8) the participant asks the researcher if the participant could continue the study if the participant has to quit in an hour, the participant could come back

7) the participant tries to find the name of the options menu in the menu document

2) the participant concludes that the name is the action overflow button
7) the participant searches for “android action overflow button”
7) the participant looks through a set of StackOverflow answers
7) the participant finds and answer for how to set the overflow menu and copies it
4) the participant pastes the copied method into the Activity onCreate method but finds that it doesn’t compile
7) the participant goes back to looking at the answer and finds that one of the comments is that the answer is a hack
7) the participant concludes that the answer may be for an old version
7) the participant searches again for “android three dot icons” and selects the next answer about how to add the icon to a button
3) hypothesis: could the icon line be placed in the xml to make the application work
    4) the participant the answer line “showAsAction: never” to the activity main xml to the LinearLayout.
    7) the participant tests the application and finds that the buttons do not appear
7) the participant goes back to the question
4) the participant copies another answer and tries to paste it into the xml - specifically the line about the vertical ellipses for strings
2) the participant mentions that the participant tries the answers in order but also prioritizes the simpler fixes over the more complex ones (in reality only seems to try the relatively simple ones)
7) the participant goes to the third stack overflow page and sees another answer about the overflow menu - creating the getOverflowMenu
7) the participant copies the method
7) the participant then searches for the getOverflow method in the Activity
3) would adding the method to the Activity fix the problem?
    4) the participant adds the method and then adds a call to the method in the onCreate for the activity
    7) the participant tries to run the app but the app fails because of the ellipses string line that was added earlier.
    4) the participant removes the ellipses string line
    7) the participant runs the application but finds that that the buttons still do not appear
    (28:01)
    7) the participant goes back to reading the answer and finds that it links to a similar method
    3) hypothesis: is the problem the location that getOverflowMenu is called?
    4) the participant moves the getOverflowMenu call earlier in the onCreate method.
    7) the participant runs the app but doesn’t see anything
    3) hypothesis: the getOverflowMenu method is not executing
        7) the participant adds a debugging statements to the method and then runs the application in debugging mode

7) the app doesn't seem to reach the debugging point so the participant sets an debugging point at the getOverflowMenu call and finds that the app calls the method and executes the method
2) the participant comments that getOverflowMenu is executed successfully
7) the participant finds that the options menu doesn't show
5) the participant tries to change strategies

7) the participant tries to remove as much as possible other than the getOverflowMenu to make sure nothing is interfering with it to make sure that the getOverflowMenu isn't having negative interactions with the rest of the code
4) the participant comments our the program after the getOverflowMenu
7) the participant runs the program and still doesn't see the options menu
4) the participant comments out the calls before the getOverflowMenu statement in the onCreate method.
7) the participant runs the application again but still doesn't see the options menu
8) the participant has to go so the researcher and participant stop the task. The participant estimates that it will take another 30 minutes

Participant 14, Task 2, Fault 29
7) the participant reads through the prompt to understand the problem
7) the participant runs the application to see the crash and read through the stack trace
4) the participant reads the warning message:

```
The onCreateOptionsMenu should be
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  MenuInflater inflater = getMenuInflater();
  Inflater.inflate(R.menu.game_menu, menu);
  return true;
}
```

4) the participant remove the parameter from the constructor
4) the participant notices that appStartTime is now undefined so the participant sets the value to getArguments() (based on his understanding of the warning) but that doesn't work
7) the participant rereads the message and determines that the participant would have to set and get the Bundle
7) the participant changes to OtherMetadataFragment and reads the error message again
7) the participant Google's "java subclass with parameters" and clicks on the first StackOverflow answer
7) the participant copies a generic class declaration
4) the participant starts to change the OtherMetadataFragment class to use generics (11:33)

5) the participant realizes the answer was for a function and not a class after trying to implement the answer
7) the participant goes back in the web browser and clicks on the third link about subclassing
7) the participant scrolls through the answers but then decides to go back after none of the answers were promising
7) the participant clicks on the fifth answer and finds generics in a class definition
4) the participant changes the OtherMetadataFragment class to successfully use generics (13:30)
3) the participant asks the researcher if the code is fixed and the researcher points out that the red underlines in the IDE mean that the app will not compile
7) the participant looks though the code to see the problems in the files
7) the participant goes back to looking up subclass examples online but he isn't happy with any of the links
7) the participant looks up "java subclass extends class warning" and clicks on the first link (a StackOverflow question on "Subclassing a Java Builder Class"
2) the participant states that he is looking for an answer that looks similar to his situation
5) the participant reverts the changes he had made to the app
5) the researcher asks the participant why the participant reverted the app and the participant said that the participant still thinks the problem is a generics problem but the participant wanted to see the original error message again.
7) the participant Google searches for "java the fragment should provide a default constructor" and clicks on the first StackOverflow answer
2) the participant exclaims that this problem seems to be the same problem that the participant has
7) the participant notices an answer that says the participant can disable IDE warnings to remove the warning
7) the participant reads through other answers
7) the participant comments that the participant may be able to avoid implementing a fix with generics
7) the participant goes back to reading other answers to look for possible fixes
4) the participant tries to comment out the failing constructor but runs into other warnings
7) the participant looks into another StackOverFlow question about default contructors
7) the participant notices that we haven't called a super function in our constructor but then finds that super is called in other methods.
2) this StackOverFlow answer leads the participant to believe that the participant needs to change the XML file.
7) the participant looks the fragment_other_metadata.xml to see if there is anything the participant needs to change
5) the participant doesn't notice any changes to make in the file

7) the participant Googles “android ignore warning” and clicks on the StackOverFlow question “Suppessing Lint Warnings”
7) the participant finds that there is an Annotation SuppressLint that removes warnings
3) Can the participant fix the warning problem with a suppress warning annotation annotation?

- 4) the participant copies the SuppressLint statement from online into the code but the warning is still in the code
- 7) the participant Googles “java supess lint warnings”
- 7) the participant finds a “SupressWarning annnoation”
- 2) the participant deduces that the participant has to use a parameter to the suppress warning call
- 7) the participant reads through the code and the online example to figure out what the parameter should be
- 4) the participant copes the unchecked parameter from the example
- 5) the warning was still there
- 7) the participant starts looking through multiple online site to read about SuppressWarning
- 2) the participant concludes there may be a suppress warning catch all
- 7) the participant Google searches “java suppress warnings all” and clicks on the Stack overFlow question
- 7) the participant finds and answer for Eclipse that says to use “all” as the parameter
- 4) the participant changes the parameter to all and notices that the warnings are all gone (28:44)

Participant 14, Task 5, Fault 63
(the coding for this task was lost somehow - I guess I typed it into Google drive when I didn’t have internet connection and didn’t realize that the results never synced)

Participant 15, Task 3, Fault 17

7) the participant reads through the prompt to understand the problem
7) the participant runs the application to see the crash and read through the stack trace
3) hypothesis: is the options menu is hidden somewhere in the layout code?

- 7) the participant navigates through the xml files, looking for the default starting screen and the other layouts
- 5) the participant glances through all the different files

7) the participant sees the onCreateOptionsMenu call
2) the participant concludes that the options menu is created but not linked to the program
7) the participant looks through the file and Activity for an incorrect to linking the options menu

2) the participant concludes that the options menu is not being linked in the fragment **(very similar to the directive)**
7) the participant mentions that the participant was looking for how to add the options menu in the code but changes his mind
7) the participant decides to look into the onCreateOptionsMenu where the inputs are set to see if that provides any more information
7) the participant Google searches “options menu android”
2) the participant mentioned that he decided to search because it would hopefully be faster than trying to implement the correct way to add the menu himself.
7) the participant looks into the developer menu page
4) the participant commented out menu.clear in case that was preventing the menu from appearing (23:20)
7) the participant ran the application and found that the menu is not displaying
7) the participant decides to look into the visibility fragment to see if it contains a way to set the visibility of the options menu
7) the participant searches for options in the file but doesn’t find anything that is helpful (only new thing a comment saying the to fix the options menu problem)
7) the participant goes back to looking through the menu document for a line to add the options menu
2) the participant mentioned that he remembered he needs to add a line to add the options menu
7) the participant looks into what the menuInflator does to understand the onOptionsMenuCreated and to understand the example code
7) the participant looks for another onCreateOptionsMenu example but doesn’t find anything
2) the participant notices that the online example of onCreateOptionsMenu is different from the method call in the code vase
3) hypotheses: is the problem that the onCreateOptionsMenu is not a standard onCreateOptionsMenu method

> 7) the participant copies the method (30:46)
> 4) the participant pastes the method in the code and notices the getMenuInflator call is broken
> 7) the participant searches for getMenuInflator to try to figure out what is setting the menu inflator
> 7) the participant reads into the documentation for onCreateOptionsMenu
> 4) the participant changes the menu call in the code
> 7) the participant seaches for “how to get menu inflator object options menu android” and clicks on the second item, the menuInflator documentation page because the first link is the menu documentation page which the developer had already been on (32:40)
> 7) the developer also opens a stack overflow page on how to get the options menu to appear and clicks on that first
> 2) the participant comments that StackOverflow is kinda hit or miss

7) the participant sees an answer with getMenuInflator
3) is the problem that menu inflator is not being set correctly?
7) the participant searches “get menu inflator android”
4) the participant tries to change the line to MenuInflator.getMenuInflator()
7) the participant looks into the MenuInflator class and tries to find the getMenuInflator method, but doesn’t find the method after searching
7) the participant finds a reference in a comment saying getMenuInflator is an Activity method
7) the participant finds the method
2) the participant realizes that the Activity has a menuInflator but the Fragment does not
7) the participant searches for “fragment android” to better understand what the Fragment does
7) the participant clicks on the Fragment developer page but quickly clicks off of it when he finishes thinking through his justification - realized he already knew what that page would tell him
2) the participant questions if he could add the options menu in the activity page
7) the participant searches “options menu android in fragment” and clicks on the Android developer Fragment page again when that is the top link
7) the participant also selects a StackOverflow answer and the top question contains the directive **a setHasOptionsMenu(true)** call
7) the participant reads another answer with the directive in it
4) is the onCreateOptionsMenu was correct?
7) the participant goes back to the code a decides to add back the old onCreateOptionsMenu call, because the participant realizes that the other onCreateOptionsMenu call was for Activities
4) the participant clearly focuses on an answer with the directive. The participant copies the directive line in the onCreate method
4) the participant comments out the menu.clear statement but then uncomments it
7) the participant adds a debug statement to see if the app even reaches the onCreateOptionsMenu code and runs the application to finds that the method is called
7) the participant debugs to check if the getMenuInflator is called and finds that it is
7) the participant runs the application normally and finds that it works

The participant does another task but the video cuts off halfway through for some reason