1. Figure out what the blue in the UI for quest state is trying to show with blue.
   1. I went to the screen for qa (quests and adventures) and guessed that the class named QuestTable was building that output
   2. It has a method createQuestLabel which colors the quest title blue if it’s state is “triggered”



1. So now I wonder if the quest state “available” means anything at all . . . so I searched for its references
   1. This is where I see it:
      1. in the tests for the client
      2. in the sequence tests
      3. in QuestManager and QuestState in the server
      4. in the tests of GameShared
   2. Clearly, I need to investigate only the things in iii above
      1. QuestManager is making things available if we pass their start date! NOW I REMEMBER WHY WE HAVE AVAILABLE!!!!! They can stay hidden until we get to a particular part of the semester. I’ll go update the WIKI so we don’t forget that again!
      2. Wait! What state are they in before they become available? NULL!!! That’s not my favorite, but what that really means is that there is no entry in the quest state table until the quest becomes available. I can live with that. I figured that out by looking at the place in the code where they become available (QuestManager.getQuestStateByID). Uhh – that means that we change their state as a side effect of searching for them. GROSS!!
      3. I’ll make the wiki match this, and I will document that side effect as something we should clean up in the issue list
2. Given that the blue in the quest list means triggered, how were we seeing the two example quests as triggered before we did anything?
   1. At this point, I just tried playing the game to see if I could recreate what Courtney was showing us.
   2. As soon as I logged in as Nick, both quests triggered. I wonder why?
   3. I looked at the quest state enum to see where nick starts – he has no initial quest states, so everything would be hidden? That doesn’t seem to make sense.
   4. So, the question is, what is causing the quests to trigger? I looked at QuestsForTest to see how those quests were defined and they have a gamelocation which means they should be triggered when we walk on that location. Mystery Solved! That is why they are triggering in the game and not the tests!
   5. I can prove that by nulling that game location in the enum, rebuilding the db, and restarting the game. They should no longer trigger by location . . .
   6. Ok – making it null broke the builder . . . The builder is looking for the two parts of a location – map name and position. Since the location is null, looking for its parts is broken. I will make each of those parts return null if the location itself is null.
   7. The create constructor on the row data gateway also crashed on null. Need to write a test and make sure we can store null.
   8. That actually took some work, but now trigger locations can be null for the tests. Let’s see if that breaks the game . . .
   9. Nothing seems to be broken. Now I can trigger the first quest (talk to the NPC tutor) by stepping on the right square. And I can complete that quest by using local chat with the tutor. However, that doesn’t trigger anything . . .
3. So, why doesn’t the next quest get triggered?
   1. My first attempt to see what is happening is to look at the code for completing a quest as that is where I expect to see the next quest get triggered.
   2. I see the code I expected to find in the complete method of QuestState. Let’s put a breakpoint at the start of that method and see what is happening
   3. That code seems to be working and sending a report that the player needs to be notified of the change in state. I wonder if that message is getting to the client?
   4. Wait! When run without a breakpoint, the quest list array is throwing a concurrent modification exception:

Exception in thread "Timer-2" starting to read from Socket[addr=/127.0.0.1,port=65054,localport=1874]

java.util.ConcurrentModificationException

at java.util.ArrayList$Itr.checkForComodification(Unknown Source)

at java.util.ArrayList$Itr.next(Unknown Source)

at model.QuestManager.handlePlayerChatCriteriaCompletion(QuestManager.java:285)

at model.QuestManager.receiveReport(QuestManager.java:213)

at model.QualifiedObservableConnector.sendReport(QualifiedObservableConnector.java:80)

at model.ChatManager.sendChatToClients(ChatManager.java:65)

at model.ChatManager.processChatMessage(ChatManager.java:84)

at model.CommandChatMessageReceived.execute(CommandChatMessageReceived.java:38)

at model.ModelFacade$ProcessCommandQueueTask.run(ModelFacade.java:92)

at java.util.TimerThread.mainLoop(Unknown Source)

at java.util.TimerThread.run(Unknown Source)

* 1. That means that we are trying to add something to the list while we are walking through it. This is now a pretty gnarly bug to fix . . . Here is a more detailed description of what is happening:
     1. When the QuestManager sees that someone tried to chat with the NPC tutor, it is getting the list of all of the quest states for that player in the QuestManager.
     2. We are walking through that list to see if any quests were completed by that chat.
     3. When we see the quest complete (correctly), we try to trigger the new quest.
     4. That modifies the list of quest states for that player in the QuestManager, but that’s the list we are walking through – there is our concurrent modification (modifying while iterating)
  2. I think we need to make a copy of the quest state list and walk through that. Let’s try it . . . That got rid of the concurrent modification exception. Now I see the quest state change message showing the second quest being triggered getting to the client. Why isn’t it getting to the GUI?
  3. I see everything going into the model of the client correctly. Then it gets processed in sendQuestStateChangeReport of ThisClientsPlayer. However, look at this code:

**for** (ClientPlayerQuestStateDTO q : questList)

{

**if** (q.getQuestID() == questID)

{

System.***out***.println("!!!!!!!");

q.setState(newState);

QuestStateChangeReport r = **new** QuestStateChangeReport(**this**.getID(),

q.getQuestID(), questDescription, newState);

QualifiedObservableConnector.*getSingleton*().sendReport(r);

}

}

* 1. This means that we are going to report changes of state if they are in quests we know about. Well that clearly isn’t good enough.
  2. I proved this was the root of the problem by making Nick start on a slightly different position. When he walks onto the right position, the server triggers the NPC tutor quest, but that never shows up on the client UI.
  3. This is a general bug! For all prior quests, you trigger them by landing on the point you teleport into a room. That means that you are going to get the quest status as part of connecting to the server.