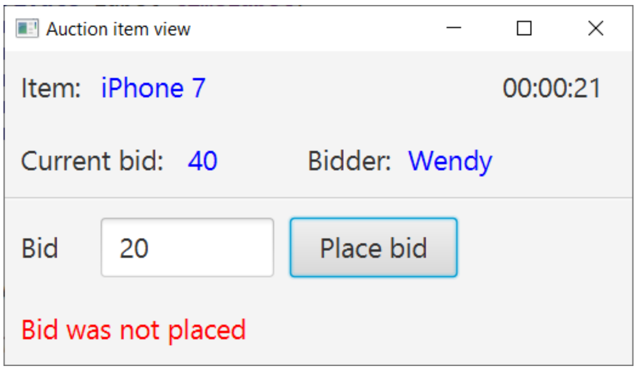
Exercise: MVVM - Auction

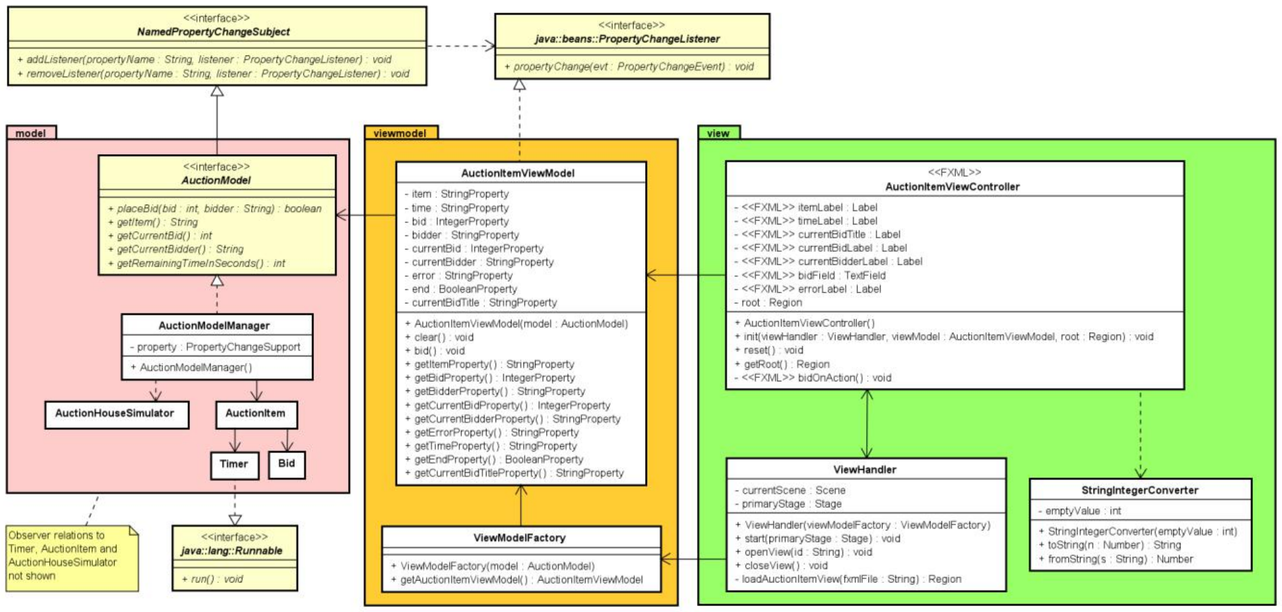
The purpose of this exercise is to create an MVVM application to simulate an auction, see below.

Desired behaviour:



- The window show the auction item, and constantly updating the current bid and current bidder (observer)
- The Timer is constantly updated (observer)
- You can place a bid (using "You" as the bidder name)
- You are not allowed to place a bid (and an appropriate message is shown)
  - after the auction is ended
  - if your bid is not higher than the current bid
  - if the current bidder is also You
- When the auction is ended, the title "Current bid" is changed to "Final bid" and the background colour of the timer becomes red (`timeLabel.setStyle("-fx-background-color:RED")`). *Alternatively, deactivate the button (requires the button to have an fx:id in the FXML file and the same button as an instance variable in the ViewController. The statement to deactivate the button is e.g.* *`bidButton.setDisable(true)`*)

Your job is to implement the two classes related to the window (classes `AuctionItemViewModel` and `AnctionItemViewController`) such that you end up having an MVVM application exactly as shown in the class diagram below.


<<interface>>
NamedPropertyChangeSubject
+ addListener(propertyName : String, listener : PropertyChangeListener) : void
+ removeListener(propertyName : String, listener : PropertyChangeListener) : void
<<interface>>
java::beans::PropertyChangeListener
+ propertyChange(evt : PropertyChangeEvent) : void
model
<<interface>>
AuctionModel
+ placeBid(bid : int, bidder : String) : boolean
+ getItem() : String
+ getCurrentBid() : int
+ getCurrentBidder() : String
+ getRemainingTimeInSeconds() : int
AuctionModelManager
- property : PropertyChangeSupport
+ AuctionModelManager()
AuctionHouseSimulator
AuctionItem
Timer
Bid
Observer relations to Timer, AuctionItem and AuctionHouseSimulator not shown
<<interface>>
java::lang::Runnable
+ run() : void
viewmodel
AuctionItemViewModel
- item : StringProperty
- time : StringProperty
- bid : IntegerProperty
- bidder : StringProperty
- currentBid : IntegerProperty
- currentBidder : StringProperty
- error : StringProperty
- end : BooleanProperty
- currentBidTitle : StringProperty
+ AuctionItemViewModel(model : AuctionModel)
+ clear() : void
+ bid() : void
+ getItemProperty() : StringProperty
+ getBidProperty() : IntegerProperty
+ getBidderProperty() : StringProperty
+ getCurrentBidProperty() : IntegerProperty
+ getCurrentBidderProperty() : StringProperty
+ getErrorProperty() : StringProperty
+ getTimeProperty() : StringProperty
+ getEndProperty() : BooleanProperty
+ getCurrentBidTitleProperty() : StringProperty
ViewModelFactory
+ ViewModelFactory(model : AuctionModel)
+ getAuctionItemViewModel() : AuctionItemViewModel
view
<<FXML>>
AuctionItemViewController
- <<FXML>> itemLabel : Label
- <<FXML>> timeLabel : Label
- <<FXML>> currentBidTitle : Label
- <<FXML>> currentBidLabel : Label
- <<FXML>> currentBidderLabel : Label
- <<FXML>> bidField : TextField
- <<FXML>> errorLabel : Label
- root : Region
+ AuctionItemViewController()
+ init(viewHandler : ViewHandler, viewModel : AuctionItemViewModel, root : Region) : void
+ reset() : void
+ getRoot() : Region
- <<FXML>> bidOnAction() : void
ViewHandler
- currentScene : Scene
- primaryStage : Stage
+ ViewHandler(viewModelFactory : ViewModelFactory)
+ start(primaryStage : Stage) : void
+ openView(id : String) : void
+ closeView() : void
- loadAuctionItemView(fxmlFile : String) : Region
StringIntegerConverter
- emptyValue : int
+ StringIntegerConverter(emptyValue : int)
+ toString(n : Number) : String
+ fromString(s : String) : Number

Model:

- The full model is given
- The model uses the Observer design pattern with the interface `NamedPropertySubject` (also given) and `PropertyChangeListener` from the java.beans library. There are three property names used (and you may add a listener to these the following way)
    - `model.addListener("bid", this)` to get notified when a new bid is placed
    - `model.addListener("time", this)` to get timer notification every second
    - `model.addListener("end", this)` to get notified when the auction ends

  or simply
    - `model.addListener(this)` to get notified for all events
- The model has one item, a timer (a thread) and a simulation with two bidders Bob and Wendy placing bids every 15 seconds.

ViewModel:

- Implement the two classes `AuctionItemViewModel` and `ViewModelFactory` as shown in the class diagram
- Look at the list of desired behaviour when implementing methods `bid` and `propertyChange`

View:

- Implement class `AuctionItemViewController` (the FXML file `AuctionItemView.fxml` is also given)
    - Look at the list of desired behaviour when making the binding in method `init`
    - You may use the given class `StringIntegerConverter` when making the binding between a labels property and an `IntegerProperty` from the ViewModel.
- Class `ViewHandler` is also given

Class implementing Application and a Main method:

- Both classes are also given