# Design decisions

1. Use a MVC architecture, since the application is inherently interactive, contains a model, a controlling part and several views.   
   With this architecture we want to decouple views from the model and the controller logic.  
   This will help us to achieve flexibility and reuse. More specific, if we take a new controller, we do not have to modify the model or view part. Also, when using a new GUI library, we can replace the view package and reuse the controller and model package. For this last reuse example, there is one problem left: the model elements inherit from the DefaultMutableTreeNode class. This class
2. Make class Model a singleton, so that both the View classes as well as the Controller classes can directly access the model (without having to pass all references).   
   Further, we want to enforce that only one model can exist in our application.  
     
   The reason for having only one model is that in the current assignment-setup there is also only one diagram pane to draw the packages and classes belonging to one model. For a future extension, one could create multiple diagram panes and also multiple models.
3. Reference to ModelController form the views. TODO: should we make this singleton (or how should we access ModelController from multiple views)
4. TODO: ModelController as it gives a single point of access to change the model, and a single point of responsibility to call notifyObservers.