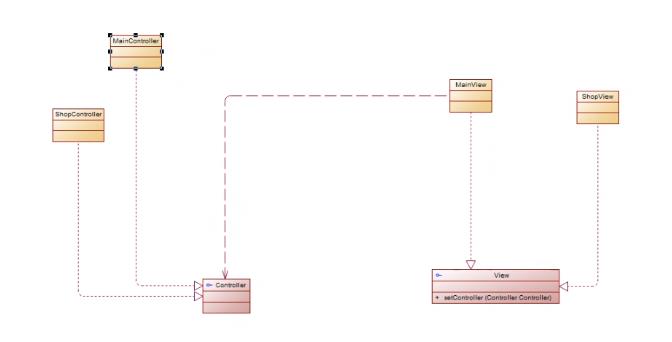
# ATdIT To do

Refresh Methode direkt in der View aufrufen  
 ActionListener Klassen pro Funktion (ggf. parametisieren)  
 Lambda Funktion (ActionEvent mitgeben) -> Bsp.: buttonRefresh.addActionListener  
 -> Lesbarer, kein Absprung in einer andere Klasse, man muss keinen ActionListener extra implementieren und Kopplung ist weg  
 Eig.: Controller kennt View (wir haben´s umgekehrt, starke View)  
 -> Hat zur Folge, dass die View nicht austauschbar ist  
 -> Controller in Interface, Instanziierung vom Controller durch Factory-Methode  
 -> müssen wir noch machen, ist ein ActionItem im Bewertungsbogen  
 -> mehr Interfaces benutzen  
  
 dependency inversion, open closed principle, sorted list  
  
 Exceptions behandeln:  
 Enduser sieht Konsole nicht: In der Fehlermeldung muss die Art der Exception im Pop Up stehen, ggf. Stacktrace

Runtime Exceptions müssen nicht abgefangen werden  
 setDefaultExceptionHandler, UncaughtExceptionHandler  
 Alert (Pop Up) abbilden  
 Fehler in Log Datei schreiben  
 Fehlerklassen immer weiter nach oben geben, bis es Sinn ergibt (z.B. in´s UI)

Build-Tool bei Tests (Maven z.B.)  
 Einen Assert pro Unit Test  
 View Klasse nicht in Tests instanziieren



inputIsValid kein guter Methodenname  
 inputIsvalid Methode auslagern in etxra Klasse (checkNewPasswordValid als Methodenname oder so), Validator  
 -> if´s auflösen in verschiedene Methoden  
 -> Methoden Exceptions werfen lassen (nicht return true oder false)  
 -> gehört nicht in den Controller  
 -> nicht so viele returns, am Besten genau ein Ausstiegspunkt in der Methode am Ende  
  
 Zu Design-Änderungen wie z.B. die starke View am BEsten ein paar Worte schreiben  
  
 Internationalisierung mit properties Dateien, am Besten auch maven  
 -> ResourceBundle.getBundle("Pfad") ohne das .properties  
 .setLanguage(), Sprache kann man in Project Configurations ändern

* Controller sollte nicht die eigentlichen Objekte der View kennen
* AddJobController > kennt die textfelder
* View setDat, get Data und das selber macht

* SCHNITTSTELLEN

* AWT Textfeld im Controller, man kann den Controller nicht entkoppeln

* Schnittstellen definieren

* View sollten interface implementiert

* View kennt den Controller, besser wären da ACtionHanlder events

* Screen setOnActionSave

* Umgekehrt view ruft nicht methode des controller aufrufen

* Controller ruft View methode auf, setzt ActionListener für View

* View sagt knopf gedrückt

* Lamda methoden, methoden referenz

* Klassen sollen nichts voneinander wissen

* Meinetwegen auch mit interfaces

* KLlassen entkoppeln

* View ändern heißt bei uns auch Controller ändern

* Kriegen Action Items von Britta

* Logik kapseln > Prozessklasse

* ER Diagramm darstellen

* Fremdschlüssel auf eine Rolle/JobTitle User ist Farmer Kunde Arbeitssuchender

* Die rollen aufdröseln

* UML – Diagramm > unter Diagramme erstellen Übergeordnete klassendiagramme -> Schnittstellen Definitionen Hirachie usw. private Attribute nicht interessant

* Internationalisierung > Text nicht Hartcodieren

* Massendeklarationen am Anfang untereinander ausführlich

* Error Handling > Pop-up Datenbank verbindung ist nicht da dann terminieren

* Connection einfach immer mit getConnection

Fragen:

Was machen wir mit der Main Klasse?

Wie macht man dass das View den Controller nicht kennt. Wir dieser einfach nur als ActionListener registriert?

<https://github.com/wurstbox/ibait_2020_mvc>

* Maven 1-2 P. Luca
* Scope mind. 3 P. Vortreffen (Mittwoch)
* MVC Vorteffen
* MVC Luis, Martin, Sophia (Mittwoch)
* Exceptions 1-2 P. Martin, Johanna
* Tests mind. 2 P. Sophia, Luis
* Internationalisierung 1-2 P. Johanna, Martin
* Diagramme 1-2 P. Luca
* Das Lambda Zeug 1-2 P. Sophia, Luis
* Kleinigkeiten alle lol
* Wissenschaftliche Ausarbeitung -> Bezug zu Mutterstadt?
* Präsentation 1-2 P. Luca, Johanna