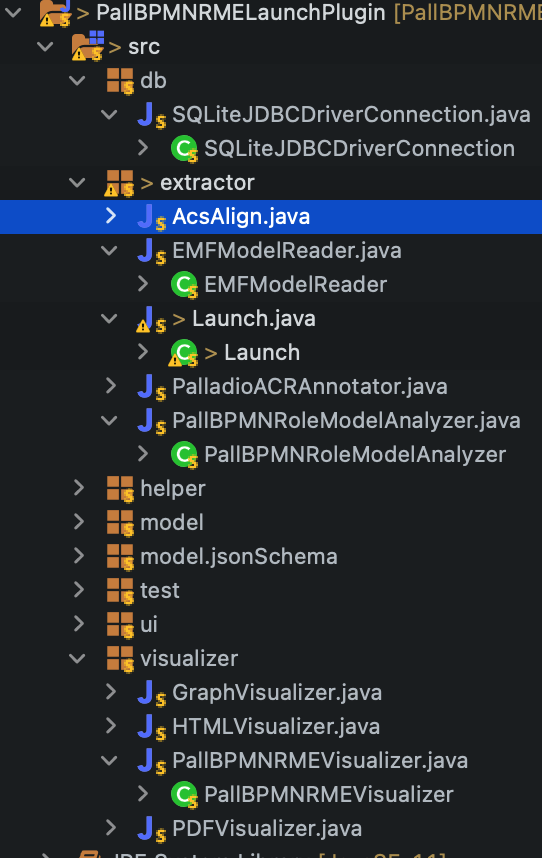
**PallBPMNRME Methoden-Kontrollfluss**

Übersicht Projekt Ordnerstruktur mit den verwendeten Paketen und Klassen



## Kontrollfluss der einzelnen Methoden - Extraktion aus Prozessen & Befüllung der Datenbank

extractor.Launch: lauch()

1. Wird gestartet beim Ausführen des Plugins
2. Im RunAs Configuration Fenster müssen die folgenden Modelle ausgewählt sein:
3. Bpusagemodel, organizationenvironmentmodel, datamodel, system, repository, poolextension intbiisl\_lp, elscToAs intbiis\_lp
4. Datenbank wird initialisiert und Tabellen erstellt
5. Für jeden BPUsageModelpfad wird aus dem Modell ein Szenario Behaviour extrahiert
6. Für jedes SzenarionBehaviour wird extractor.PallBPMNRoleModelAnalyzer:extractModelsFromScenarioBehaviour(scenarioBehaviour, db, null) aufgerufen
   1. ScenariaBehaviour – ist das entsprechende ScenarioBehaviour Klassen Objekt
   2. Db ist das initialisierte Datenbankobjekt
   3. Null ist der Prozess der im Falle eines Loop oder Branch beim Aufruf gesetzt wird

* Wenn Actor Step -> Extrahiere alle erforderlichen Daten/Objekte und schreibe sie in die Datenbank:
  + As = ActorStep
  + BusinessActivity = ActorStep.getEntityName() - Name des Actor Steps
  + Role = getOrganizationalUnit(role.getEntityName()) + ":" + role.getEntityName(); - zusammengesetzte Rolle aus dem Namen der Oragnizational unit und dem Rollenname
  + BusinessPermission = getDataObjects(as); - gibt die Datenobjekte des Actor Steps zurück
  + BusinessPermissionIds = getDataObjectsIds(as); - gibt die DatenobjektIds des Actor Steps zurück
  + ISDataType = getRepositoryDataTypes(as); - gibt die Datentypen des Actor Steps aus dem Repository zurück
  + Kombiniert Read und Write Elemente:
    - BusinessPermission = combineReadWriteElements(dbBusinessPermission);
    - BusinessPermissionIds = combineReadWriteElements(dbBusinessPermissionIds);
  + ArrayList<String> acrbData = new ArrayList<>(); init um alle extrahierten Werte zu speichern
  + AcrbData.add(alle vorher gesetzten Werte) fügt alle Werte in die ArrayList
  + db.insertACRBPDataToTable(acrbData); schreibt die Werte in die ACRBP Tabelle
* Wenn ELSC -> Extrahiere alle erforderlichen Daten/Objekte und schreibe sie in die Datenbank:
  + EntryLevelSystemCall elsc = (EntryLevelSystemCall) a; - castet Action zu ELSC
  + dbISActivity = elsc.getEntityName(); - ISActivity entspricht dem Namen des ELSC
  + dbSystem = elsc.getProvidedRole\_EntryLevelSystemCall().getProvidingEntity\_ProvidedRole().getEntityName(); - System entspricht dem Namen der provided role des ELSC
  + operat = elsc.getOperationSignature\_\_EntryLevelSystemCall(); - enthält die Parameter und Rückgabetypen (READ/WRITE Datentyp)
  + dbServiceCall = elsc.getProvidedRole\_EntryLevelSystemCall().getEntityName() + "." + operat.getEntityName() + "()"; - setzt sich aus dem provided role Namen und dem Funktionsnamen zusammen. Wenn keine FUnktion vorhanden dann ist zweiter Teil null
  + matchingRole = matchingActorStepForELSC.getResponsibleRole().getEntityName(); - entspricht dem Actor Step zu dem die ELSC gehört
  + elscMatchingASData.add(alle vorher gesetzten Werte) - schreibt alle extrahierten Werte in eine ArrayList<String>
  + db.insertACRWorkflowDataToTable(elscMatchingASData); - schreibt alle Werte in die ACRWorkflow Tabelle
* Wenn Loop-> Get Body (Szenario Behaviour) des Loops und rufe extractModelsFromScenarioBehaviour(loopScenarioBehaviour, db, dbProcess); auf
* Wenn Branch-> Get branchTransitions (Szenario Behaviour) des Branchs, iteriere über diese und rufe für jedes extractModelsFromScenarioBehaviour(branchScenarioBehaviour, db, dbProcess); auf
* Iteriert wiederholt über alle Actions um passende ADR/RDR für die Actor-Steps zu finden
  + Wenn ReleaseDeviceResourceAction -> Extrahiere alle erforderlichen Daten/Objekte und schreibe sie in die Datenbank:
    - ArrayList<Integer> rdrMatchingIds = getMatchingId("rdr", dbProcess, rdra.getEntityName()); - gibt alle aus dem Prozess extrahierten Ids von Actor Steps zurück die den RDR enthalten
    - RdrMatchingASData.add(ActorStepId, rdra Name, rdra Berechtigung) - schreibt für jede Id einen Eintrag in die ArrayList<String>
    - db.insertACRAcquireReleaseDataToTable(rdrMatchingASData); - schreibt die Einträge in die ACRAcquireRelease Tabelle
  + Wenn AcquireDeviceResourceAction -> Extrahiere alle erforderlichen Daten/Objekte und schreibe sie in die Datenbank:
    - ArrayList<Integer> adrMatchingIds = getMatchingId("adr", dbProcess, adra.getEntityName()); - gibt alle aus dem Prozess extrahierten Ids von Actor Steps zurück die den ADR enthalten
    - AdrMatchingASData.add(ActorStepId, adra Name, adra Berechtigung) - schreibt für jede Id einen Eintrag in die ArrayList<String>
    - db.insertACRAcquireReleaseDataToTable(adrMatchingASData); - schreibt die Einträge in die ACRAcquireRelease Tabelle

## Generierung und Visualisierung der Modelle

1. Am Ende von launch wird der PallBPMNRMEVisualizer initialisiert und dann
2. visualizer.visualizeDB(); aufgerufen
   * HTMLVisualizer htmlVisualizer = new HTMLVisualizer(projectPath); - init des HTML Visualizer zum Ausgeben der einzelnen HTML Tabellen
   * Danach wird für jede Ausgabe hier am Role Model dargestellt folgendes aufgerufen:
     1. htmlVisualizer.writeToHTML("Role Model", getRoleModelFromDb());
        1. ArrayList<String[]> roleModel = new ArrayList<String[]>();
        2. String[] roleModelHeader = { "Id", "Role", "BusinessPermission", "ISDataType" };
        3. roleModel.add(roleModelHeader);
        4. Abrufen der entsprechenden Daten per Query aus der Datenbank und speichern/zurückgeben in roleModel