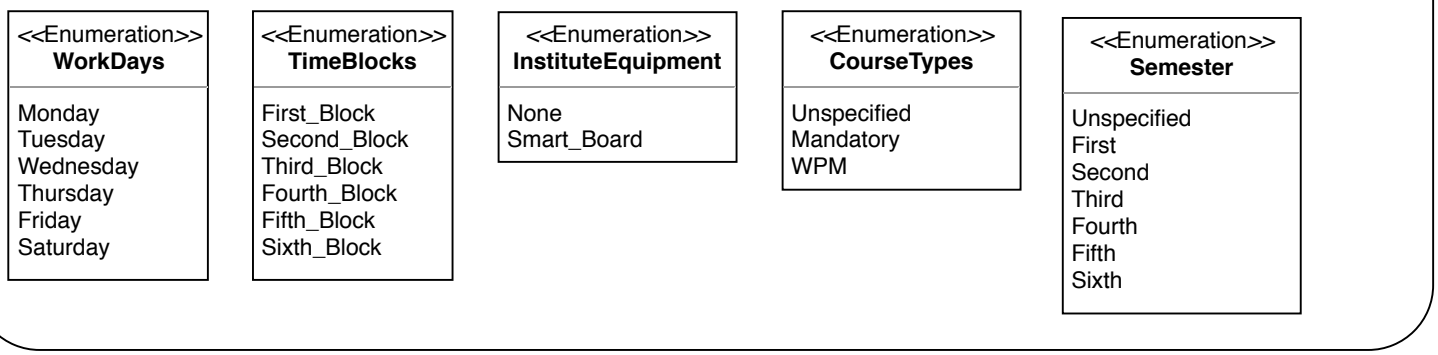
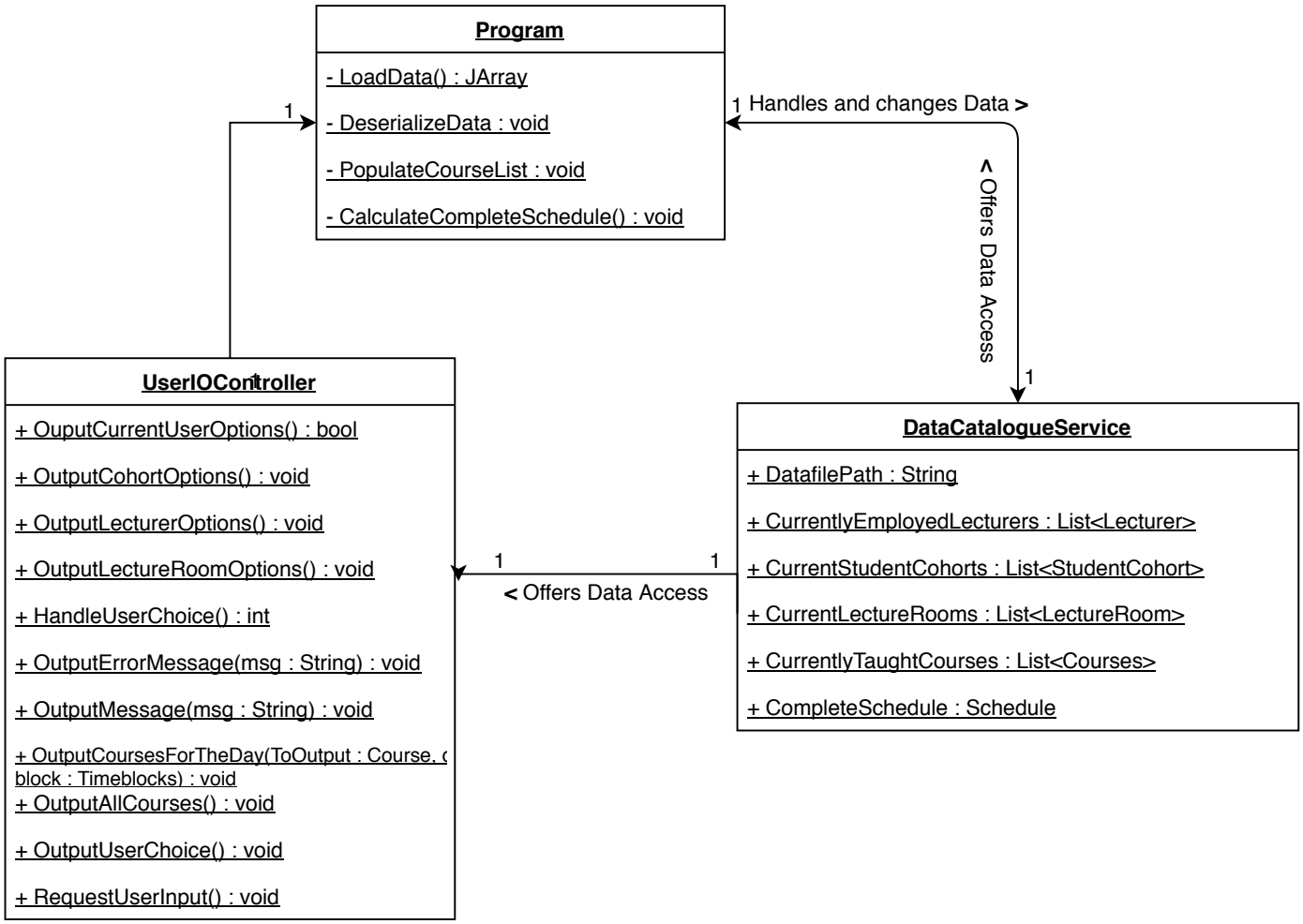
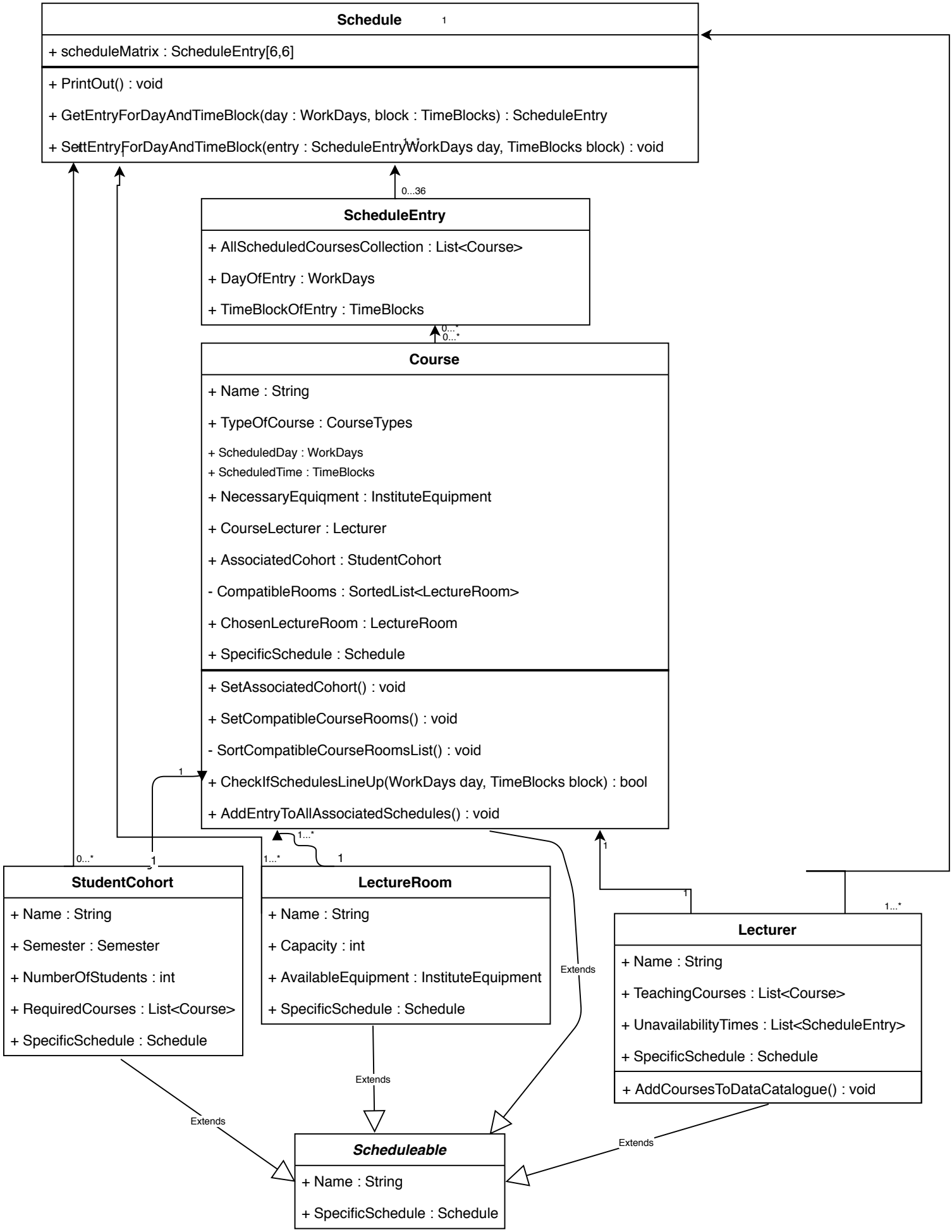
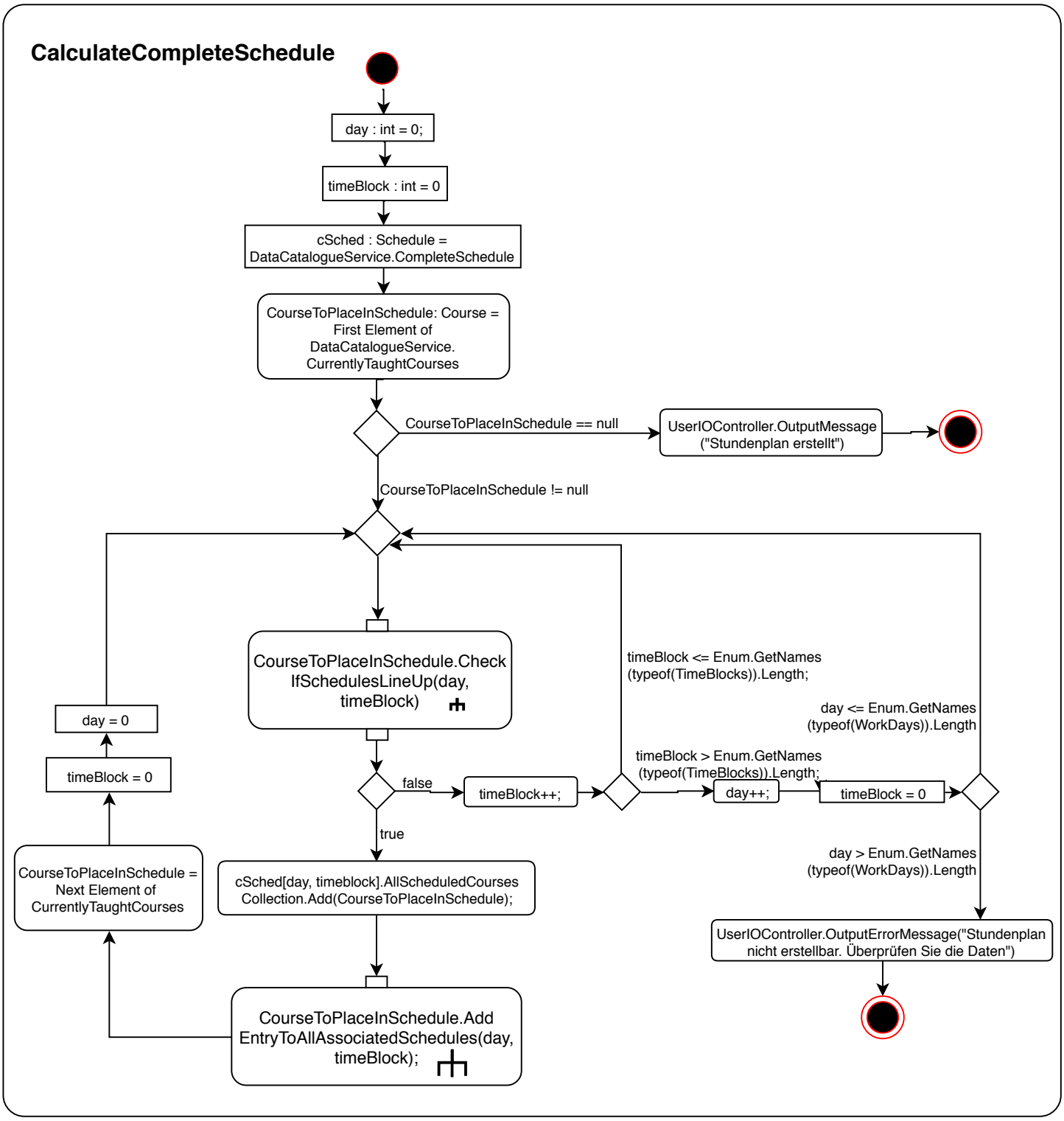
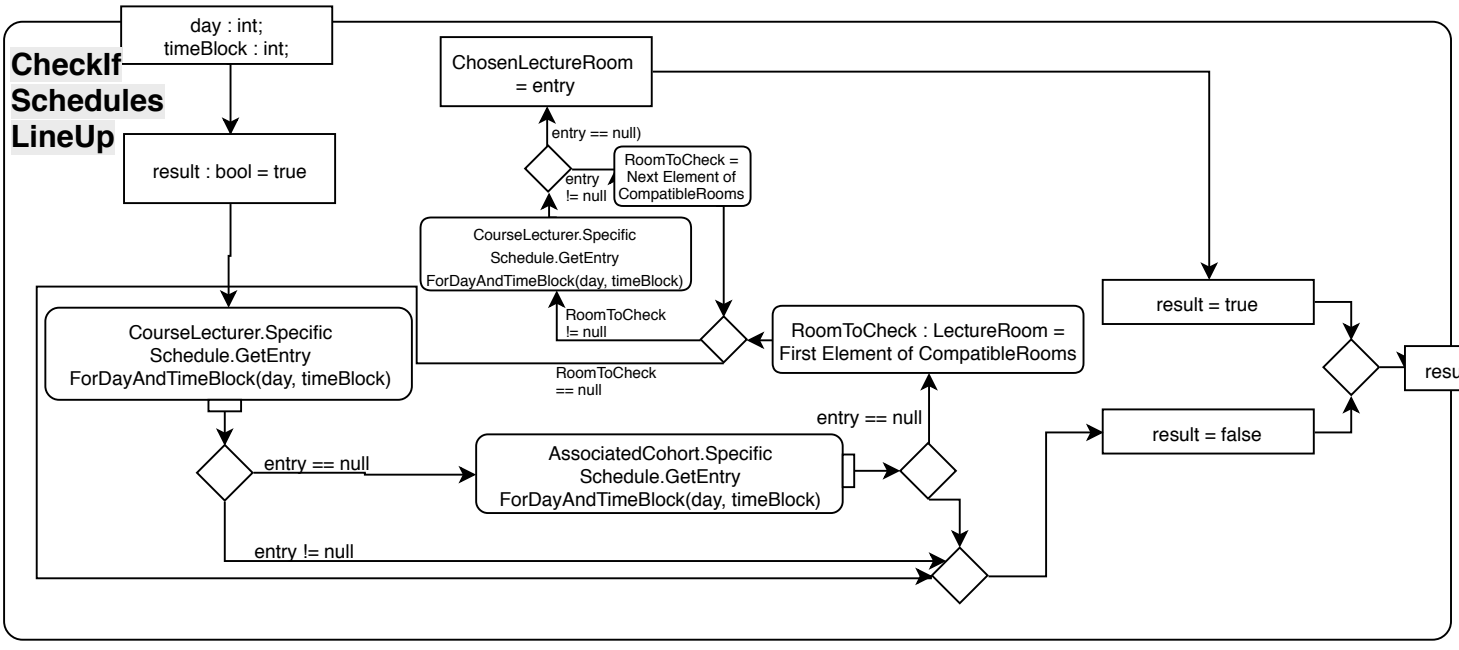
Global Enumerators

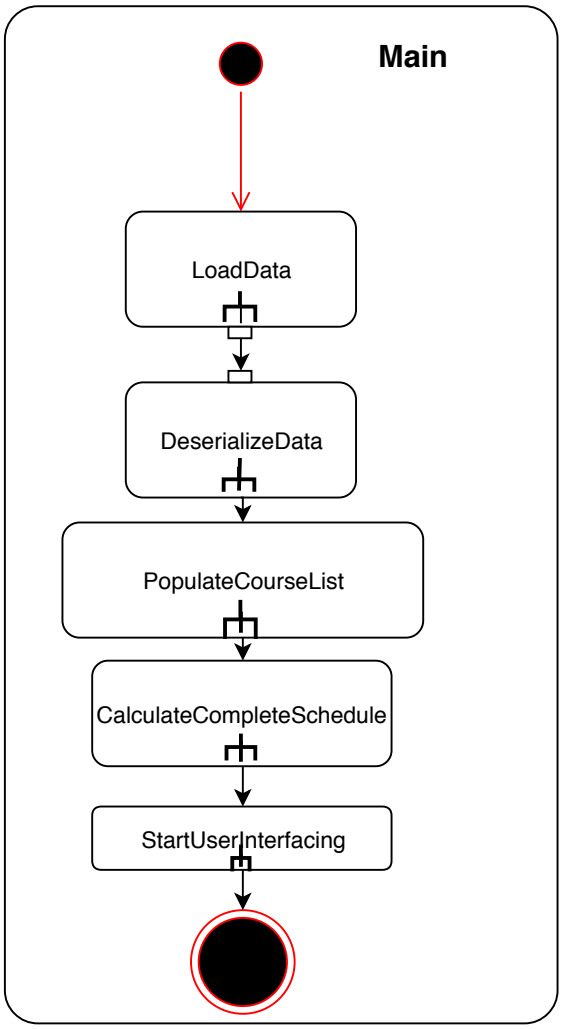


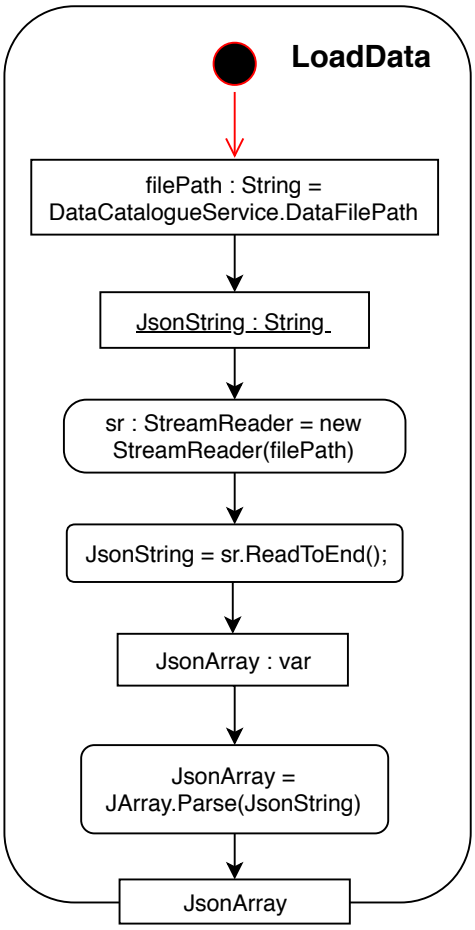


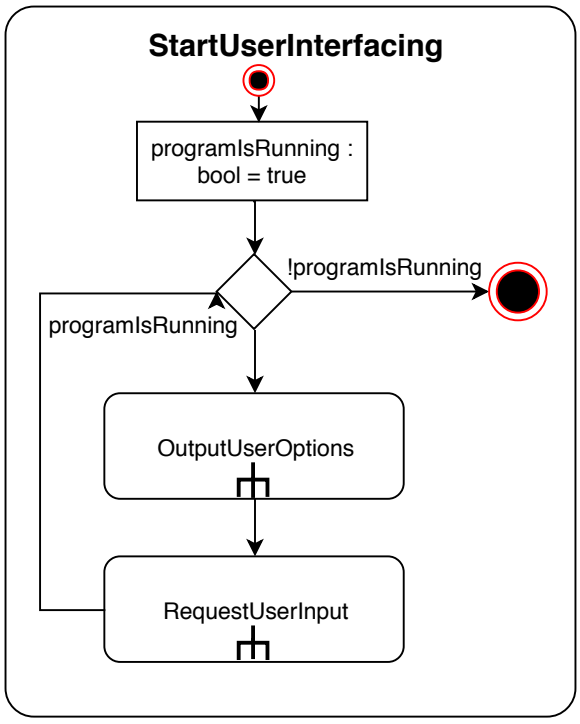

Schedule
+ scheduleMatrix : ScheduleEntry[6,6]
+ PrintOut() : void
+ GetEntryForDayAndTimeBlock(day : WorkDays, block : TimeBlocks) : ScheduleEntry
+ SettEntryForDayAndTimeBlock(entry : ScheduleEntry, WorkDays day, TimeBlocks block) : void
1
1
0...36
ScheduleEntry
+ AllScheduledCoursesCollection : List<Course>
+ DayOfEntry : WorkDays
+ TimeBlockOfEntry : TimeBlocks
0...*
0...*
Course
+ Name : String
+ TypeOfCourse : CourseTypes
+ ScheduledDay : WorkDays
+ ScheduledTime : TimeBlocks
+ NecessaryEquiqment : InstituteEquipment
+ CourseLecturer : Lecturer
+ AssociatedCohort : StudentCohort
- CompatibleRooms : SortedList<LectureRoom>
+ ChosenLectureRoom : LectureRoom
+ SpecificSchedule : Schedule
+ SetAssociatedCohort() : void
+ SetCompatibleCourseRooms() : void
- SortCompatibleCourseRoomsList() : void
+ CheckIfSchedulesLineUp(WorkDays day, TimeBlocks block) : bool
+ AddEntryToAllAssociatedSchedules() : void
1
1...*
1...*
1
0...*
1
1
1
1...*
StudentCohort
+ Name : String
+ Semester : Semester
+ NumberOfStudents : int
+ RequiredCourses : List<Course>
+ SpecificSchedule : Schedule
LectureRoom
+ Name : String
+ Capacity : int
+ AvailableEquipment : InstituteEquipment
+ SpecificSchedule : Schedule
Lecturer
+ Name : String
+ TeachingCourses : List<Course>
+ UnavailabilityTimes : List<ScheduleEntry>
+ SpecificSchedule : Schedule
+ AddCoursesToDataCatalogue() : void
Extends
Extends
Extends
Extends
Scheduleable
+ Name : String
+ SpecificSchedule : Schedule

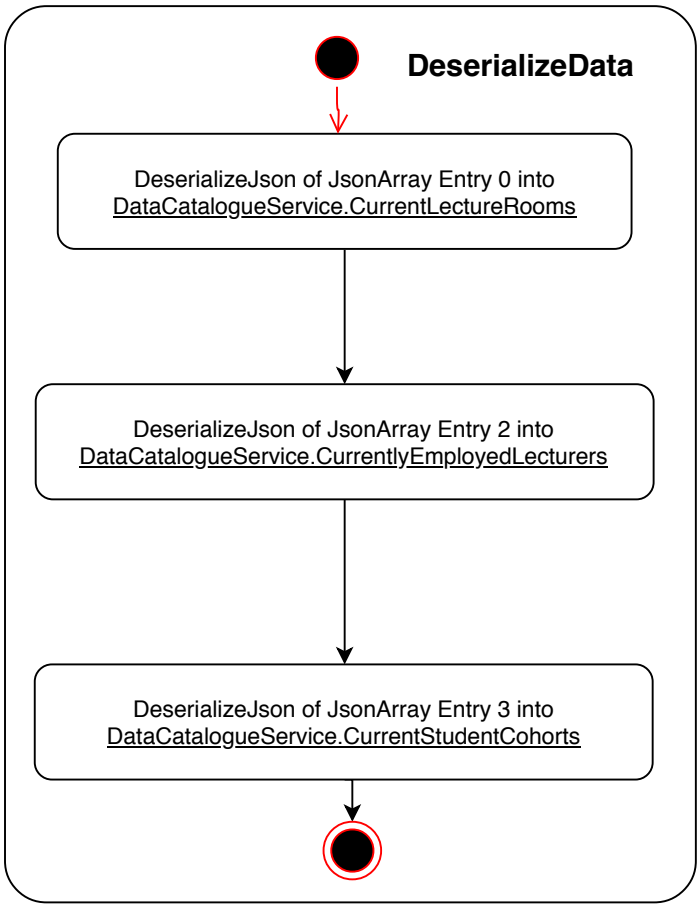

CalculateCompleteSchedule
day : int = 0;
timeBlock : int = 0
cSched : Schedule = DataCatalogueService.CompleteSchedule
CourseToPlaceInSchedule: Course = First Element of DataCatalogueService. CurrentlyTaughtCourses
CourseToPlaceInSchedule == null
UserIOController.OutputMessage ("Stundenplan erstellt")
CourseToPlaceInSchedule != null
CourseToPlaceInSchedule.Check IfSchedulesLineUp(day, timeBlock)
timeBlock <= Enum.GetNames (typeof(TimeBlocks)).Length;
day <= Enum.GetNames (typeof(WorkDays)).Length
timeBlock > Enum.GetNames (typeof(TimeBlocks)).Length;
day = 0
timeBlock = 0
false
timeBlock++;
day++;
timeBlock = 0
true
CourseToPlaceInSchedule = Next Element of CurrentlyTaughtCourses
cSched[day, timeblock].AllScheduledCourses Collection.Add(CourseToPlaceInSchedule);
day > Enum.GetNames (typeof(WorkDays)).Length
UserIOController.OutputErrorMessage("Stundenplan nicht erstellbar. Überprüfen Sie die Daten")
CourseToPlaceInSchedule.Add EntryToAllAssociatedSchedules(day, timeBlock);

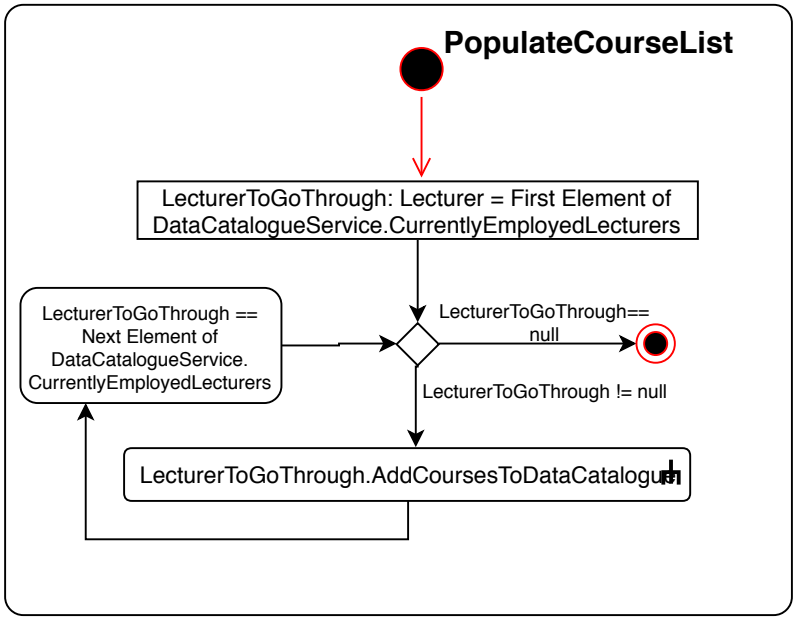

CheckIf Schedules LineUp
day : int;
timeBlock : int;
result : bool = true
ChosenLectureRoom = entry
entry == null)
entry != null
RoomToCheck = Next Element of CompatibleRooms
CourseLecturer.Specific Schedule.GetEntry ForDayAndTimeBlock(day, timeBlock)
RoomToCheck != null
CourseLecturer.Specific Schedule.GetEntry ForDayAndTimeBlock(day, timeBlock)
RoomToCheck == null
RoomToCheck : LectureRoom = First Element of CompatibleRooms
result = true
result
entry == null
result = false
entry == null
AssociatedCohort.Specific Schedule.GetEntry ForDayAndTimeBlock(day, timeBlock)
entry != null

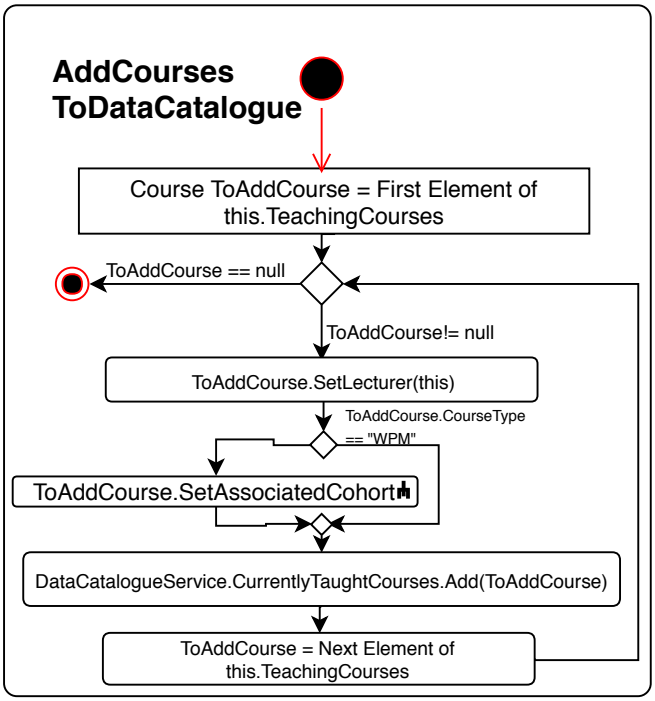

Main
LoadData
DeserializeData
PopulateCourseList
CalculateCompleteSchedule
StartUserInterfacing

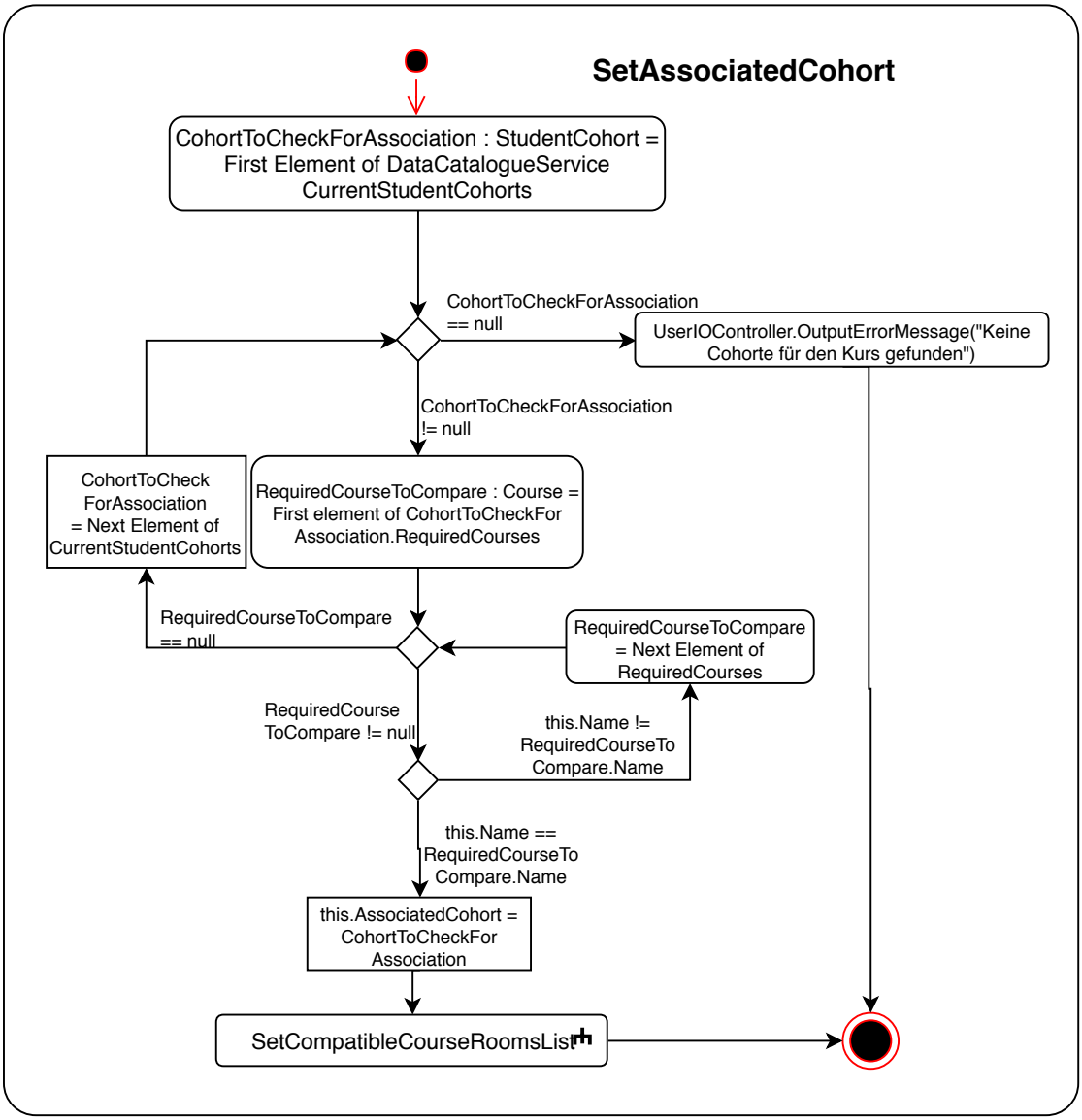

LoadData
filePath : String = DataCatalogueService.DataFilePath
JsonString : String
sr : StreamReader = new StreamReader(filePath)
JsonString = sr.ReadToEnd();
JsonArray : var
JsonArray = JArray.Parse(JsonString)
JsonArray

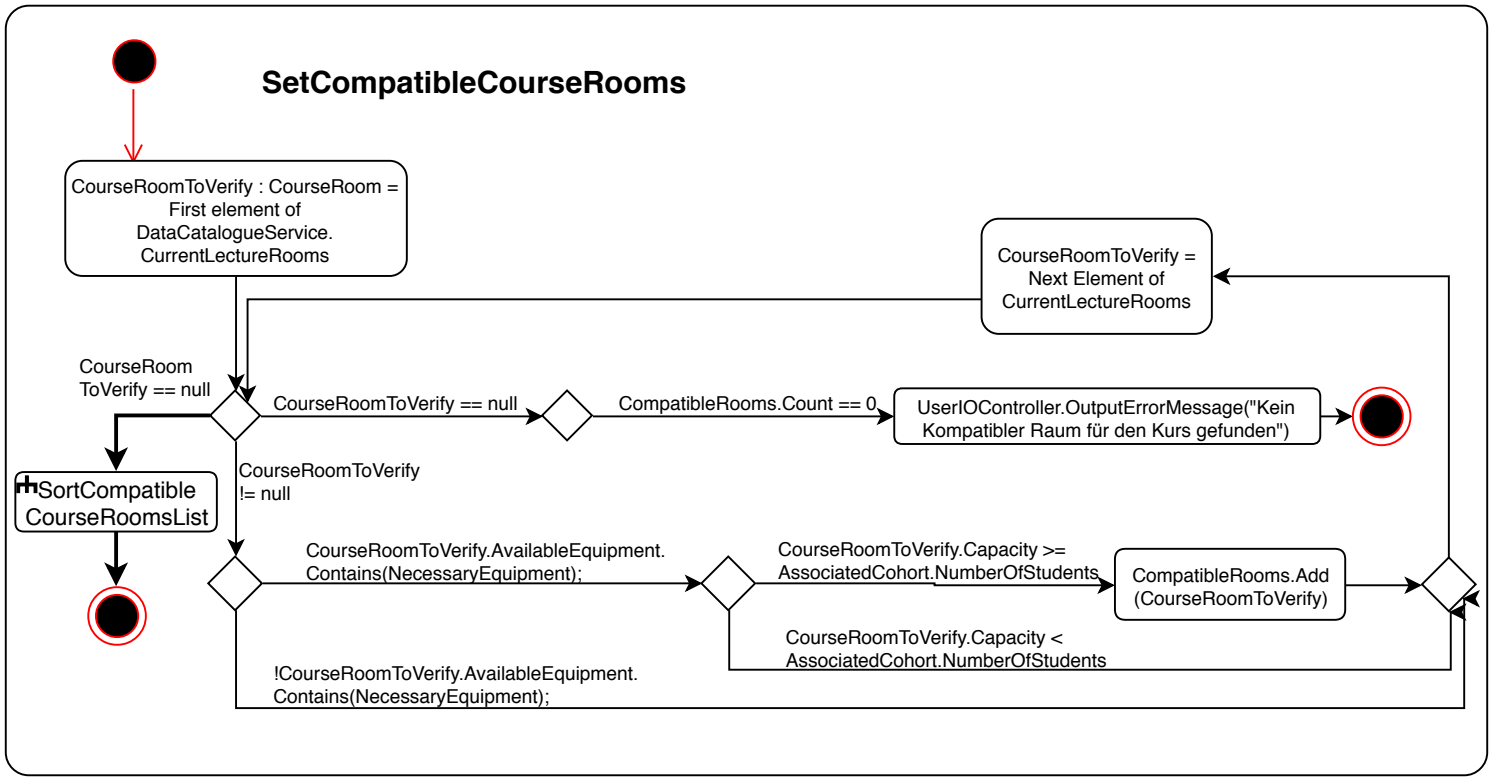

StartUserInterfacing
programIsRunning : bool = true
!programIsRunning
programIsRunning
OutputUserOptions
RequestUserInput

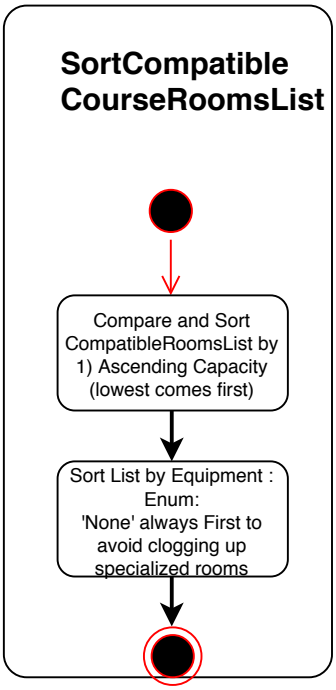

DeserializeData
DeserializeJson of JsonArray Entry 0 into DataCatalogueService.CurrentLectureRooms
DeserializeJson of JsonArray Entry 2 into DataCatalogueService.CurrentlyEmployedLecturers
DeserializeJson of JsonArray Entry 3 into DataCatalogueService.CurrentStudentCohorts


PopulateCourseList
LecturerToGoThrough: Lecturer = First Element of DataCatalogueService.CurrentlyEmployedLecturers
LecturerToGoThrough == Next Element of DataCatalogueService. CurrentlyEmployedLecturers
LecturerToGoThrough== null
LecturerToGoThrough != null
LecturerToGoThrough.AddCoursesToDataCatalogue


AddCourses ToDataCatalogue
Course ToAddCourse = First Element of this.TeachingCourses
ToAddCourse == null
ToAddCourse!= null
ToAddCourse.SetLecturer(this)
ToAddCourse.CourseType == "WPM"
ToAddCourse.SetAssociatedCohort
DataCatalogueService.CurrentlyTaughtCourses.Add(ToAddCourse)
ToAddCourse = Next Element of this.TeachingCourses


SetAssociatedCohort
CohortToCheckForAssociation : StudentCohort = First Element of DataCatalogueService CurrentStudentCohorts
CohortToCheckForAssociation == null
UserIOController.OutputErrorMessage("Keine Cohorte für den Kurs gefunden")
CohortToCheckForAssociation != null
CohortToCheck ForAssociation = Next Element of CurrentStudentCohorts
RequiredCourseToCompare : Course = First element of CohortToCheckFor Association.RequiredCourses
RequiredCourseToCompare == null
RequiredCourseToCompare = Next Element of RequiredCourses
RequiredCourse ToCompare != null
this.Name != RequiredCourseTo Compare.Name
this.Name == RequiredCourseTo Compare.Name
this.AssociatedCohort = CohortToCheckFor Association
SetCompatibleCourseRoomsList


SetCompatibleCourseRooms
CourseRoomToVerify : CourseRoom = First element of DataCatalogueService. CurrentLectureRooms
CourseRoomToVerify = Next Element of CurrentLectureRooms
CourseRoom ToVerify == null
CourseRoomToVerify == null
CompatibleRooms.Count == 0
UserIOController.OutputErrorMessage("Kein Kompatibler Raum für den Kurs gefunden")
SortCompatible CourseRoomsList
CourseRoomToVerify != null
CourseRoomToVerify.AvailableEquipment. Contains(NecessaryEquipment);
CourseRoomToVerify.Capacity >= AssociatedCohort.NumberOfStudents
CompatibleRooms.Add (CourseRoomToVerify)
CourseRoomToVerify.Capacity < AssociatedCohort.NumberOfStudents
!CourseRoomToVerify.AvailableEquipment. Contains(NecessaryEquipment);


SortCompatible CourseRoomsList
Compare and Sort CompatibleRoomsList by 1) Ascending Capacity (lowest comes first)
Sort List by Equipment : Enum: 'None' always First to avoid clogging up specialized rooms