Verslag: Takehome 2

Literatuurstudie

Bij exam timetabling zie je vaak de volgende variabelen:

- Examenperiode: Een Datum- en tijdsperiode waarin het examen wordt gegeven.
- Lokalen: Een lokaal met een bepaalde capaciteit. Hierin kunnen er ook nog verschillende variabelen zijn zoals: Een lokaal die niet gebruikt mag worden of liefst niet gebruikt wordt.
- Studenten: Een student kan niet op hetzelfde tijdstip op meerdere plekken zijn. Een soft constraint is vaak ook de lengte tussen twee examens.

Er kunnen ook verschillende voorkeuren zijn bij een examen zoals:

- Een voorkeur voor wanneer een examen wordt gehouden.
- Een voorkeur in welk lokaal een examen wordt gehouden.
- Examens die in verschillende/dezelfde ruimtes moeten gegeven worden.
- Examens die in verschillende/dezelfde tijdsloten moeten gegeven worden.
- Een volgorde van examens
- …

Elke school heeft hiervoor wel verschillende parameters.

Framework

Ik heb gekozen voor Optaplanner.

Hoe heb ik mijn oplossing gemodelleerd?

Ik maak gebruik van de console om mijn oplossing weer te geven, deze heb ik geordend per Timeslot.

Welke zetten heb ik geïmplementeerd?

Ik heb een extra classe ExamTable aangemaakt om de verschillende examens en tijdsloten bij te houden. Deze wordt aangemaakt in de DataReader.

```java
@PlanningSolution
public class ExamTable {
    @ProblemFactCollectionProperty
    @ValueRangeProvider(id = "timeslotRange")
    private List<TimeSlot> timeslotList;
    @PlanningEntityCollectionProperty
    private List<Exam> examList;
    @PlanningScore
    private HardSoftScore score;

    public ExamTable() {
    }

    public ExamTable(List<TimeSlot> timeslotList, List<Exam> examList) {
        this.timeslotList = timeslotList;
        this.examList = examList;
    }

    public List<TimeSlot> getTimeslotList() { return timeslotList; }

    public List<Exam> getExamList() { return examList; }

    public HardSoftScore getScore() { return score; }
}
```

Daarna stel ik de SolverFactory in met een aangemaakte constraint provider.

```
SolverFactory<ExamTable> solverFactory = SolverFactory.create(new SolverConfig()
        .withSolutionClass(ExamTable.class)
        .withEntityClasses(Exam.class)
        .withConstraintProviderClass(ExamTableConstraintProvider.class)
        .withTerminationSpentLimit(Duration.ofHours(1)));
```

In de constraint provider geef ik de hard- en soft constrains op met behulp van optaplanner. Deze kan dan opgelost worden door de solver. Daarna print ik de oplossing in de console.

Hoe heb ik de kostfunctie geïmplementeerd?

Deze is geïmplementeerd met optaplanner. Er is een HardSoftScore veld in de ExamTable. Bij de constraint provider worden de scores toegevoegd als er een regel wordt verbroken. Een Hard score als er een conflict is tussen studenten en een Soft score wanneer studenten examens na elkaar hebben. Deze is vermenigvuldigd met een getal die afhangt hoeveel dagen er tussen zitten.

```
return constraintFactory
        .forEach(Exam.class) UniConstraintStream<Exam>
        .join(Exam.class, Joiners.equal(Exam::getTimeslot)) BiConstraintStream<Exam, Exam>
        .filter(((exam1, exam2) -> {
            if(exam1.getID() == exam2.getID()){
                return false;
            }
            List<Integer> list1 = exam1.getSID();
            List<Integer> list2 = exam2.getSID();
            Set<Integer> set = new HashSet<>(list1);
            return set.removeAll(list2);
        }))
        .penalize( constraintName: "Student conflict", HardSoftScore.ONE_HARD);
```

```
return constraintFactory
        .forEach(Exam.class) UniConstraintStream<Exam>
        .join(Exam.class) BiConstraintStream<Exam, Exam>
        .filter(((exam1, exam2) -> {
            if(exam1.getID() == exam2.getID()){
                return false;
            }
            List<Integer> list1 = exam1.getSID();
            List<Integer> list2 = exam2.getSID();
            Set<Integer> set = new HashSet<>(list1);
            return set.removeAll(list2);
        }))
        .penalize( constraintName: "Student time between", HardSoftScore.ONE_SOFT, ((exam1, exam2) -> {
            Integer between = Math.abs(exam1.getTimeslot().getID() - exam2.getTimeslot().getID());
            if (between < 2) {
                return 16;
            } else if (between < 3) {
                return 8;
            } else if (between < 4) {
                return 4;
            } else if (between < 5) {
                return 2;
            } else if (between < 6) {
                return 1;
            } else {
                return 0;
            }
        }));
```

Resultaten

Wanneer ik de solver met een TerminationSpentLimit van een uur laat lopen krijg ik een score waarbij er geen hard constraints zijn gebroken en met een soft score van -22460.