Trabalho MC322

Sistema de distribuição de horários e salas

Guilherme Webster Chamoun 257111
Gustavo Jun Tsuji 252278
Vinicius Forato Coracin 231528

1) Resumo do sistema

O sistema é responsável por distribuir horários e salas para as aulas que ocorrerão durante um semestre em uma faculdade fictícia, inspirada na UNICAMP. Para que essa distribuição seja realizada adequadamente, não se pode alocar a mesma sala para duas aulas diferentes em um mesmo horário. Além disso, duas aulas obrigatórias de um mesmo semestre e de um mesmo curso não podem ter o mesmo horário. Por exemplo, MC322 e Cálculo III são ambas obrigatórias para o terceiro semestre de Ciência da Computação, logo, não podem ter o mesmo horário (para que assim o aluno possa assistir a ambas as aulas).

Dessa forma, passado como input uma lista de cursos (com suas respectivas matérias por semestre), uma lista de disciplinas (com suas informações - nome, lista de professores, número de créditos, etc) e uma lista de locais disponíveis, o sistema cria a grade de horários do semestre, atribuindo horários, locais e professores para as aulas.

2) Detalhes de implementação

Primeiramente, para cada curso, o sistema percorre a lista de matérias por semestre e cria a quantidade de aulas necessárias para cada uma delas. Em seguida, atribui os horários. Essa distribuição é aleatória, mas pelo modo como ela é feita, garante que não haja conflito.

Na sequência, fazemos a distribuição de salas. Para fazer isso, utilizamos um algoritmo de coloração de grafos. Criamos um grafo com uma aula em cada vértice e ligamos por uma aresta as aulas com mesmo horário. Depois, colorimos o grafo e atribuímos um local diferente para cada aula pintada com cor diferente. Caso não seja possível realizar tal distribuição (por haver número insuficiente de salas), o sistema emite um erro.

Para realizar a distribuição, o sistema leva em consideração alguns requisitos adicionais. As disciplinas possuem o tipo de espaço requerido (sala básica, sala com projetor, laboratório, auditório, etc), e possuem uma lista de possíveis institutos. Dessa forma, por exemplo, garantimos que uma aula de MC322 ocorra no CB, PB ou IC, sendo uma das aulas da semana numa sala com projetor e outra numa sala de computadores.

A princípio, pensamos que as distribuições de horário e de salas eram independentes. No entanto, isso só seria verdade se tivéssemos um número ilimitado de

salas. Se tivermos apenas 2 salas com projetor disponíveis e 3 aulas necessitarem desse tipo de sala, é evidente que não podemos deixar as três no mesmo horário. Ao invés de contornar essa dependência desenvolvendo um algoritmo que levasse isso em consideração (o que aumentaria em muito a complexidade, pois seriam duas colorações de grafo dependentes uma da outra), resolvemos colocar o algoritmo principal dentro de um loop while com try/catch. O algoritmo tenta realizar a distribuição até conseguir, ou até que o número de erros exceda um limite pré-determinado. Se o número de erros ultrapassa esse limite, um erro é lançado e a execução do sistema é interrompida.

3) Levantamento de funções

Há uma lista de cursos, locais e disciplinas pré-definidas, salvas num arquivo XML. O usuário pode optar, na interface gráfica, por remover algum desses cursos, locais ou disciplinas da distribuição. Pode, ainda, adicionar um local ou eletiva a mais. Há um botão de submit para confirmar os parâmetros escolhidos.

Após isso, o usuário pode gerar a distribuição e escolher qual curso e semestre deseja ver a grade de horários. Pode, também, visualizar a distribuição das disciplinas eletivas. Há, ainda, a opção de salvar o resultado num arquivo XML.

4) Requisitos

- **Relacionamentos:**

A composição está amplamente presente no sistema. Por exemplo, curso contém uma lista de semestres, que por sua vez contém uma lista de disciplinas. Aula contém uma disciplina e um curso associado. A composição de curso com semestre é do tipo agregação, pois a classe contida (Semester) não faz sentido se não estiver associada a um curso. Por outro lado, a relação de semestre com disciplina e de aula com disciplina e curso é associação, pois essas classes fazem sentido isoladamente.

- **Polimorfismo:**

  Polimorfismo está presente na classe abstrata Lecture e suas herdeiras MandatoryLecture e ElectiveLecture. A criação das aulas, bem como sua exposição na interface gráfica, é realizada através de métodos polimórficos.

- **Interfaces:**

  O sistema implementou duas interfaces próprias: XMLFileReader e XMLNodeReader, que são implementadas pelos arquivos Readers de XML. Além disso, utilizamos a interface Comparator, implementada em LectureComparator para ordenarmos a lista de aulas antes de mostrar os resultados.

- **Classes Abstratas:**

  Criamos duas classes abstratas: Lecture, que representa abstratamente uma aula (cujas subclasses são MandatoryLecture e ElectiveLecture), e View, cujas subclasses fazem parte da interface gráfica.

- **Interface Gráfica**
  Implementamos uma interface gráfica utilizando JavaFX. Ela é responsável por lidar com as escolhas do usuário e exibir os dados de forma interativa.

- **Design Patterns:**
  a) Singleton: Está presente em diversos pontos do código, tais como nos readers de XML e nos Views da interface gráfica. São singletons porque são classes em que não é interessante instanciar mais de uma vez.
  b) MVC: Está presente na interface gráfica, onde foi dividido a responsabilidade do controlador, da visualização e do model. Os controladores estão contidos no package Controllers, em GraphicInterface, as views estão no package Views, também de GraphicInterface, e o model é a classe LectureSelector.
  c) Strategy: Presente no package System. Dividimos a responsabilidade do algoritmo de distribuição em duas classes, ScheduleAllocator e SpaceAllocator, cujos métodos são invocados pela classe controladora AllocatorSystem. Dessa forma, dividimos a responsabilidade, facilitamos a manutenção e extensibilidade.

- **Tratamento de Exceções:**
  O sistema trata exceções. Foram definidos dois erros customizados: NoAvailableSpacesError e InsufficientSpacesError. O primeiro é lançado quando não há nenhum local de algum tipo, já o segundo quando tentamos fazer a distribuição mas não há nenhum local livre do tipo requerido.

- **Gravação e Leitura de Arquivos:**
  O sistema lê arquivos XML contendo informações dos cursos, espaços e disciplinas, e os utiliza para inicializar as instâncias dessas classes. O sistema grava as informações das aulas criadas num arquivo XML, caso o usuário desejar.

5) Estrutura do sistema (principais classes)

Pacote System:
Pacote que inclui as classes que realizam o algoritmo de distribuição

- Classe AllocatorSystem:
  - allocateSchedulesAndPlaces: invoca os métodos createLecturesWithSchedules, e assignPlaces. Caso tenha sido possível gerar uma distribuição não conflitante, mostra o resultado na interface. Senão, ele tenta novamente até conseguir ou até o número de erros gerados ultrapassar um limite pré-definido, caso em que a execução é interrompida.

- Classe ScheduleAllocator:

    - createLectures: cria as aulas eletivas e obrigatórias, evitando aulas de mesmo curso e semestre num mesmo horário. Retorna uma lista com as aulas criadas e com o horário atribuído.

- Classe SpaceAllocator:
    - assignPlaces: atribui espaços para as aulas ocorrerem de modo que duas aulas em um mesmo horário não tenham um mesmo local. Cria um grafo no qual os vértices representam as aulas e as arestas ligam aulas com mesmo horário. O grafo é, então, colorido de modo que dois vértices ligados por pelo menos um par de arestas não são coloridos com a mesma cor, ou seja, não tem uma mesma sala atribuída.

Pacote CourseRelated:
Pacote que contém classes relativas ao curso, disciplinas e aulas

- Classe Course: representa os cursos. Contém uma lista de semestres, nome, id e turno do curso.
.
- Classe Semester: representa um semestre de um curso. Contém uma lista de disciplinas e um número que representa o seu período.

- Classe Discipline: representa uma disciplina. Contém nome e ID da disciplina, número de créditos, uma lista de docentes, uma lista com os tipos de espaços requeridos e uma lista com os possíveis institutos onde ela pode ocorrer.

- Classe (Abstrata) Lecture: representa uma aula. Tem um horário associado, uma turma, respectivo docente e o espaço onde deve ocorrer.

- ClasseElectiveLecture: representa uma aula de uma disciplina eletiva

- Classe MandatoryLecture: representa uma aula de uma disciplina obrigatória. Ela contém um curso associado.

Package Space:
Pacote que contém classes relacionadas a locais de aulas

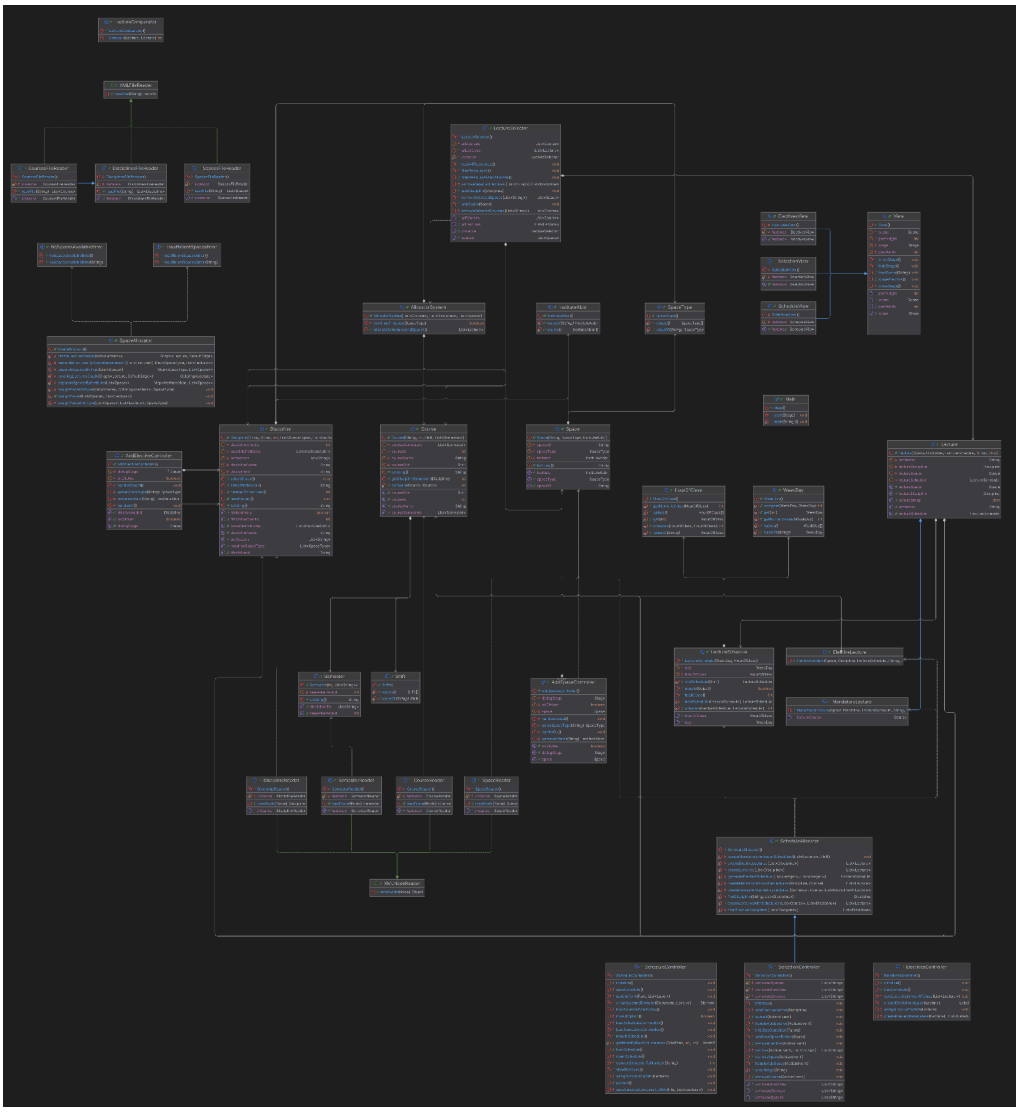
- Classe Space: representa um espaço. Contém a id, tipo do local e o instituto onde se localiza.
- Enum SpaceType: representa o tipo do local (sala básica, sala com projetor, sala de informática, laboratório, auditório, quadra etc)
- Enum InstituteAbbr: representa o instituto em que o local se localiza (CB, PB, IC, IMECC, FEEC etc)

Pacote Schedule:

Contém classes relativas aos horários

- Classe LectureSchedule: representa o horário e dia de uma aula.
- Enum WeekDay: representa o dia da semana.
- Enum HourOfClass: representa o horário de uma aula.
- Enum Shift: representa o turno de um curso (noturno ou integral)

6) UML:



Link para melhor visualização da imagem:
https://drive.google.com/file/d/1cP3TXu1tNet20RfWvvec2mMrNcp7ksek/view?usp=sharing

7) Tutorial de instalação

a) VSCode:

i) Primeiro, clone o repositório na sua IDE.

ii) Baixe a biblioteca do JavaFX em https://gluonhq.com/products/javafx/ (de preferência a versão 21.0.3).

iii) Copie o path absoluto do repositório *lib* do JavaFx e cole no arquivo launch.json, na linha de vmArgs, por exemplo:
"vmArgs": "--module-path /home/guilherme/Downloads/javafx-sdk-21.0.3/lib/ --add-modules javafx.controls,javafx.fxml".

iv) Certifique-se de que settings.json está com um único parâmetro entre colchetes, pois sua IDE deve configurar este arquivo automaticamente: "lib/**/*.jar".

b) Eclipse:

i) Primeiro, clone o repositório na sua IDE.

ii) Baixe a biblioteca do JavaFX em https://gluonhq.com/products/javafx/ (de preferência a versão 21.0.3).

iii) Clique com o botão direito na pasta do projeto e vá em Build Path -> Configure Build Path -> Java Build Path, na aba Libraries selecione Classpath -> Add External JARS e adicione os arquivos .jar presentes na pasta lib do projeto.

iv) Copie o path absoluto do repositório *lib* do JavaFx e com o botão direito na pasta do projeto, vá em Run As -> Run Configurations -> Java Application -> Main, e na aba Arguments adicione a VM arguments “--module-path /path/to/javafx-sdk-22.0.1/lib --add-modules javafx.controls,javafx.fxml”. Substitua “/path/to/javafx-sdk-22.0.1/lib” pelo path absoluto do repositório *lib* do JavaFx.