UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Aula 9: Relacionamento entre Classes [2] (Laboratório 3)

Prof. Dennis Giovani Balreira
(Material adaptado do Prof. Thiago L. T. da Silveira)

INF01120 - Técnicas de Construção de Programas

Ferramentas para diagramas UML

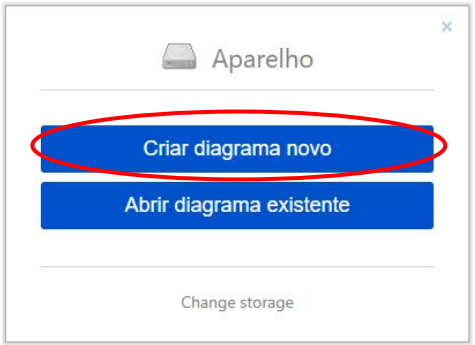
Ferramentas para UML

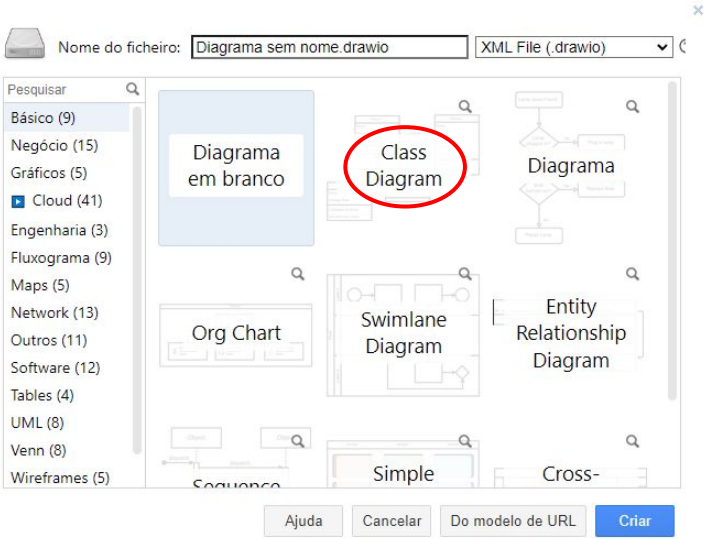
- Online:
  - Draw.io: https://app.diagrams.net/
  - Miro: https://miro.com/

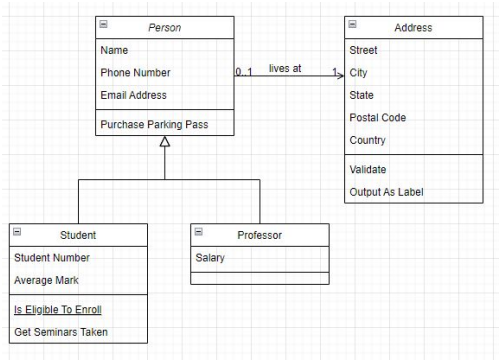
- Offline:
  - Astah: https://astah.net/products/free-student-license/
  - Star UML: https://staruml.io/

Ferramentas para UML

- Draw.io (https://app.diagrams.net/):



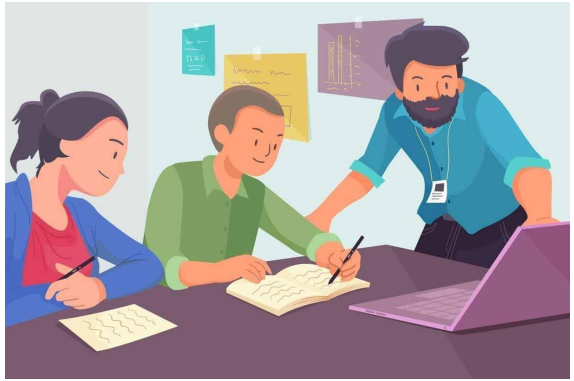




Laboratório 3

Exercício 1 - Parte 0

- A disciplina de Técnicas de Construção de Programas precisa de ajuda para ser modelada e representada por um software. Você como aluno de TCP deve auxiliar no início do desenvolvimento, nas partes de projeto e implementação.



Exercício 1 - Parte 1

- O projeto deve ter as seguintes classes:
  - `Professor`
  - `Turma`
  - `Aluno`
  - `Monitor`
  - `AplicacaoTestaUni` (não precisa ser incorporado ao diagrama)

- As classes `Professor`, `Turma`, `Monitor` e `Aluno` devem estar em um mesmo pacote chamado "`universidade`", cada uma em um arquivo .java
- A classe `AplicacaoTestaUni` deve estar no pacote "`teste`"
- **Importante:** comece pelo diagrama de classes (projeto) e só depois vá para o código (implementação)

Exercício 1 - Parte 2

- Crie um diagrama de classes que represente o problema descrito sob perspectiva do desenvolvedor e seu programa em Java, conforme instruções:
  - Crie as seguintes associações entre as classes:
    - Composição ”leciona” entre Professor (parte de) e Turma (todo)
    - Composição ”pertence” entre Aluno (parte de) e Turma (todo)
    - Agregação “auxilia” entre Monitor (parte de) e Turma (todo)

Exercício 1 - Parte 2

- Crie um diagrama de classes que represente o problema descrito sob perspectiva do desenvolvedor e seu programa em Java, conforme instruções:
  - Crie as cardinalidades conforme extraído da especificação:
    - Uma turma tem um único professor
    - Uma turma pode ter nenhum ou até três monitores
    - Uma turma pode ter de 10 a 40 alunos
    - Um professor deve lecionar pelo menos uma turma
    - Um monitor deve auxiliar apenas uma turma
    - Um aluno deve pertencer a apenas uma turma

Exercício 1 - Parte 2

- Crie um diagrama de classes que represente o problema descrito sob perspectiva do desenvolvedor e seu programa em Java, conforme instruções:
  - A classe `Professor` deve possuir os atributos `id` e `departamento` e não deve possuir métodos extras
  - A classe `Monitor` deve possuir os atributos `semestreAtual`, indicando o semestre atual do monitor e `temExperiencia`, sendo verdadeiro se já foi monitor antes ou falso caso contrário e não deve possuir métodos extras
  - A classe `Aluno` deve possuir os atributos `numeroMatricula` e `indiceDesempenho` (valor entre 0 e 10) e os métodos `aumentaIndiceDesempenho()`, `diminuiIndiceDesempenho()`, que aumentam e diminuem, respectivamente, o índice de desempenho conforme valor passado como parâmetro e o método `imprime()`, que imprime todas as informações do aluno

Exercício 1 - Parte 2

- Crie um diagrama de classes que represente o problema descrito sob perspectiva do desenvolvedor e seu programa em Java, conforme instruções:
  - A classe `Turma` deve ter os atributos `numAlunos`, `numMonitores`, `professor`, uma lista de monitores (`monitores`), uma lista de alunos (`alunos`) e os métodos `adicionaAluno()`, `removeAluno()`, que devem adicionar um aluno e remover o último aluno na lista, respectivamente, e `imprimeAlunos()`, que deve imprimir a lista de alunos atual com todas suas informações (matrícula e índice)

Exercício 1 - Parte 2

- Crie um diagrama de classes que represente o problema descrito sob perspectiva do desenvolvedor e seu programa em Java, conforme instruções:
  - Observações:
    - Todos os atributos devem ser privados
    - Crie dois métodos construtores (um sem parâmetros e outro com parâmetros) e *getters* e *setters* (se necessários) para cada classe
    - As instanciações dos elementos (comando `new`) já estão feitas na classe `AplicacaoTestaUni` (detalhada a seguir)
    - Utilize constantes para os valores mínimos e máximos em `Turma`
    - Verifique as consistências nos métodos, retornando booleanos indicando sucesso ou falha (imprima mensagens nos construtores)
    - Utilize a classe `ArrayList` para controlar as listas

Exercício 1 - Parte 3

- A classe `AplicacaoTestaUni` deve possuir o código `”AplicacaoTestaUni.java”` disponível no Moodle
  - Estude o código, observando a forma como as classes são instanciadas
  - Tente rodar o programa procurando ao máximo não alterá-lo
    - Caso precise, pode adaptá-lo em mudanças pontuais

Exercício 1 - Entrega

- Envie um único arquivo no formato **`ZIP`** contendo a implementação do **Laboratório 3 - Exercício 1** (`Professor.java`, `Monitor.java`, `Aluno.java`, `Turma.java`, `AplicacaoTestaUni.java`, **`DiagramaClasse.jpg`**)

- O nome do arquivo deve seguir o formato:

  `"l3-<primeiro_nome_do_aluno>-<último_nome_do_aluno>.zip"`

- Atente ao prazo de entrega do trabalho especificado no Moodle!

- Bom trabalho! Qualquer dúvida contate o professor

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Obrigado pela atenção!
Dúvidas?

Prof. Dennis Giovani Balreira
(Material adaptado dos Profs. Marcelo Pimenta e Thiago L. T. da Silveira)