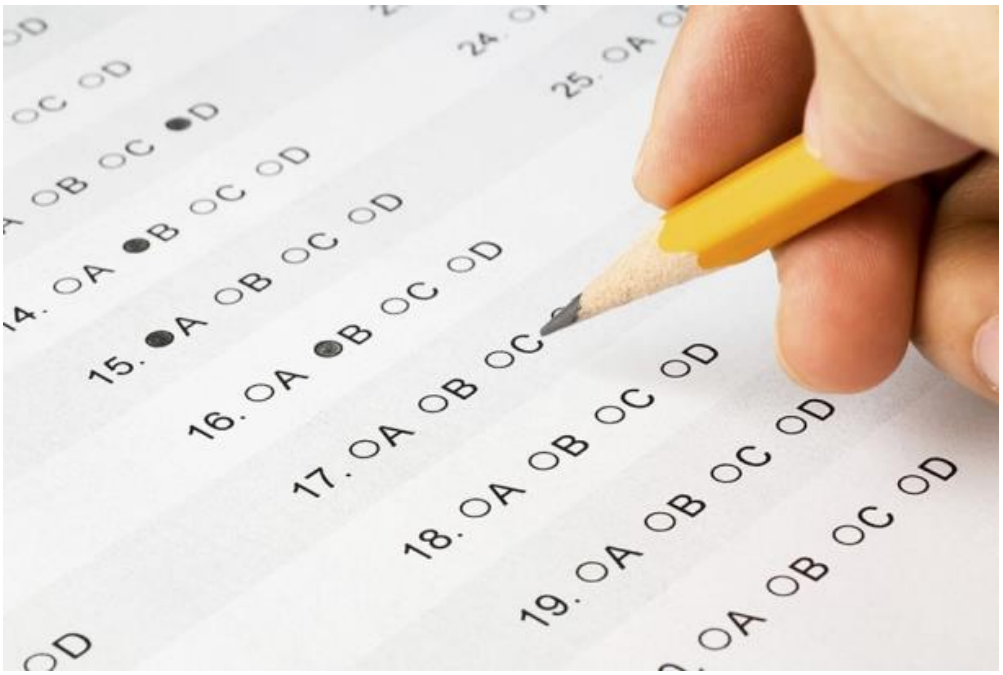
• U C •

UNIVERSIDADE DE COIMBRA

Programação Orientada a Objectos

Gestor de Exames



Bruno Manuel Leitão Grifo Nº2014228262

Fábio Silva Antunes Nº2014206491

Introdução

Este trabalho foi realizado no âmbito da cadeira de Programação orientada a objetos e tem como objetivo familiarizar os alunos para a programação orientada a objetos. O programa consiste na criação de um Gestor de Exames para ser utilizado por um administrador da Universidade de Coimbra.

Estrutura Geral

O programa não foi desenhado para ser utilizado apenas por administradores(utilizador), sendo que o administrador é uma pessoa física.

O programa tem um menu e é nessa interface que o utilizador vai trabalhar.

Menu:

1-Criar Salas
2-Criar Pessoa
3-Criar Disciplina
4-Criar Curso
5-Criar exame
6-Configurar sala exame
7-Convocar vigilantes e funcionários para um exame
8-Inscrever alunos em exame
9-Alterar/Lançar notas de um exame
10-Adicionar Docentes a uma Disciplina
11-Adicionar Alunos a uma disciplina
12-Adicionar Cursos a um aluno
13-Adicionar disciplinas a um Curso
14-Remover Pessoa
15-Remover Curso
16-Remover Curso de um Aluno
17-Remover Disciplina de um Curso
18-Remover Disciplina
19-Remover Exame
20-Remover Sala
21-Remover Pessoa de Exame
22-Remover Pessoa de Disciplina
23-Listar exames
24-Listar alunos inscritos num exame
25-Listar exames de um aluno
26-Listar docentes e funcionários de um exame
27-Listar Exames em que um docente/funcionário esta envolvido
28-Listar notas de um exame
29-Listar Disciplina
30-Listar Cursos
31-Listar Pessoas
32-Listar Salas
33-Listar cursos de um aluno
0-Sair

O programa é também constituído por 13 classes e 5 ArrayList's que são essências para que todas estas opções corram devidamente.

Classes:

- Aluno extends Pessoa: Public
- Curso: Public
- Disciplina: Public
- Docente extends Funcionario: Public
- Especial extends Exame: Public
- Exame: Abstract
- Funcionário extends Pessoa: Abstract
- GestorExames: Public
- NaoDocentes extends Funcionario: Public
- Normal extends Exame: Public
- Pessoa: Abstract
- Recurso extends Exame: Public
- Sala: Public

ArrayList's na classe principal(GestorExames):

- **ArrayList<Pessoa> lista_pessoas**: Lista onde são guardadas todas as pessoas criadas, podendo ser elas: Alunos, Docentes ou Funcionario não docentes.

- **ArrayList<Curso> lista_cursos**: Lista onde são guardados todos os cursos.

- **ArrayList<Sala> lista_salas**: Lista onde são guardadas todas as salas.

- **ArrayList<Exame> lista_exames**: Lista onde são guardados todos os exames, podendo ser eles exames: normais, de recurso ou especial.

- **ArrayList<Disciplina> lista_disciplinas**: Lista onde são guardados todas as disciplinas.

Metodos:

- public void criaSalas(): Este método vai permitir ao utilizador criar salas.
- public void criaPessoa(): Este método vai permitir ao utilizador criar pessoas(Aluno,Docente, Funcionarios não docentes).
- public void criaDisciplina(): Este método vai permitir ao utilizador criar disciplinas.
- public void criarCurso():Este método vai permitir ao utilizador criar cursos.
- public void adicionaExames():Este método vai permitir ao utilizador criar exames.
- public void configurarSalaExame():Este método vai permitir ao utilizador configurar a sala de um exame.
- public void convocaAjudantesExame():Este método vai permitir ao utilizador adicionar docente e funcionarios não docentes a um exame.
- public void increveAlunosExame():Este método vai permitir ao utilizador inscrever alunos num exame.
- public void lancaNotasExame():Este método vai permitir ao utilizador modificar ou lançar as notas de um determinado exame.
- public void adicionaDocentesDisciplina():Este método vai permitir ao utilizador adicionar docentes a uma disciplina.
- public void inscreveAlunoDisciplina():Este método vai permitir ao utilizador inscrever alunos numa disciplina.
- public void adicionaCursosAluno():Este método vai permitir ao utilizador adicionar cursos a um aluno.
- public void adicionaDisciplinasCurso():Este método vai permitir ao utilizador adicionar disciplinas a um curso.
- public void eliminaPessoa():Este método vai permitir ao utilizador eliminar uma pessoas(Aluno,Docente,Funcionarios não docentes).
- public void eliminarCurso():Este método vai permitir ao utilizador eliminar um curso.
- public void eliminaCursoAluno():Este método vai permitir ao utilizador eliminar do aluno um curso.
- public void removeDisciplinaCurso():Este método vai permitir ao utilizador remover disciplinas de um curso.
- public void selecionaDisciplinaEliminar():Este método vai permitir ao utilizador eliminar disciplinas.
- public void eliminaExame():Este método vai permitir ao utilizador eliminar exames.
- public void eliminaSala():Este método vai permitir ao utilizador eliminar salas.
- removePessoaExame():Este método vai permitir ao utilizador remover pessoas(Aluno,Docente,Funcionarios não docentes) de uma exame.
- removePessoaDisciplina():Este método vai permitir ao utilizador remover uma pessoas(Aluno,Docente,Funcionario não docente)de uma disciplina.

- public void listaExames():Este método vai permitir ao utilizador ver todos os exames.
- public void listarAlunosExame():Este método vai permitir ao utilizador ver todos os aluno de um exame.
- listaExamesAluno():Este método vai permitir ao utilizador ver todos os exames de um aluno.
- listaDocentesFuncionariosExame():Este método vai permitir ao utilizador ver todos docente e funcionário não docentes de um exame.
- public void listaExamesDocentesFuncionario():Este método vai permitir ao utilizador ver todos os exames onde um docente ou funcionario não docente esta inscrito.
- public void listaNotas():Este método vai permitir ao utilizador ver todas as notas dos alunos inscrito num exame.
- public void listaDisciplinas():Este método vai permitir ao utilizador ver todas as disciplinas.
- public void listaCursos():Este método vai permitir ao utilizador ver todos os curso.
- public void listaPessoas():Este método vai permitir ao utilizador ver todas as pessoas(Aluno,Docente,Funcionarios não docentes).
- public void listaSalas():Este método vai permitir ao utilizador ver todas as salas.
- public void listaAlunoCursos():Este método vai permitir ao utilizador ver todos os cursos de um aluno.

Aluno

Atributos:

Os atributos desta classe são: um numero_aluno(inteiro), em que cada aluno tem o seu numero distinto, um lista de cursos(HasMap<Curso,Integer>) que nos diz em que cursos o aluno está inscrito e o regime(String).

Docente

Atributos:

Os atributos desta classe não a área de investigação(String).

Curso

Atributos:

A classe curso tem como atributos o nome do curso, o grau do curso e uma lista de disciplinas. Construtor Curso(String nome_curso, int duracao_anos, String grau_curso, ArrayList<Disciplina> lista_disciplinas)

Disciplina

Atributos:

A classe disciplina tem como atributos o nome da disciplina, o docente responsável pela disciplina, outros docentes da disciplina, e uma lista de alunos inscritos na disciplina. Construtor Disciplina(String nome_disciplina, Docente docente_responsavel);

Especial

Construtor:

Especial(Disciplina disciplina, int duracao, Sala sala, Docente docente_responavel, ArrayList<Docente> vigilantes, ArrayList<NaoDocentes> apoio, HashMap<Aluno, Integer> alunos, Date exame_inicio, Date exame_final);

Exame

Atributos:

A classe exame tem como atributos uma disciplina, o docente responsável pela disciplina, outros docentes no exame, funcionários não docentes de apoio ao exame, uma lista de alunos com nota/ou sem nota lançada, uma data de inicio do exame e uma de fim, tempo de duração do exame e a sala do exame. Construtor: Exame(Disciplina disciplina, int duracao, Sala sala,Docente docente_responsavel, ArrayList<Docente> vigilantes, ArrayList<NaoDocentes> apoio,HashMap<Aluno, Integer> alunos, Date exame_inicio, Date exame_final);

Funcionario

Atributos:

A classe Funcionário tem como atributos o numero mecanográfico e a categoria. Construtor: Funcionario(String nome, String email, int numero_mecanografico, String categoria);

NaoDocentes

Atributos:

A classe NaoDocentes tem como atributos o cargo. Construtor: NaoDocentes(String nome, String email, String cargo, int numero_mecanografico, String categoria);

Normal

Construtor:

Normal(Disciplina disciplina, int duracao, Sala sala, Docente docente_responavel, ArrayList<Docente> vigilantes, ArrayList<NaoDocentes> apoio, HashMap<Aluno, Integer> alunos, Date exame_inicio, Date exame_final);

Recurso

Construtor:

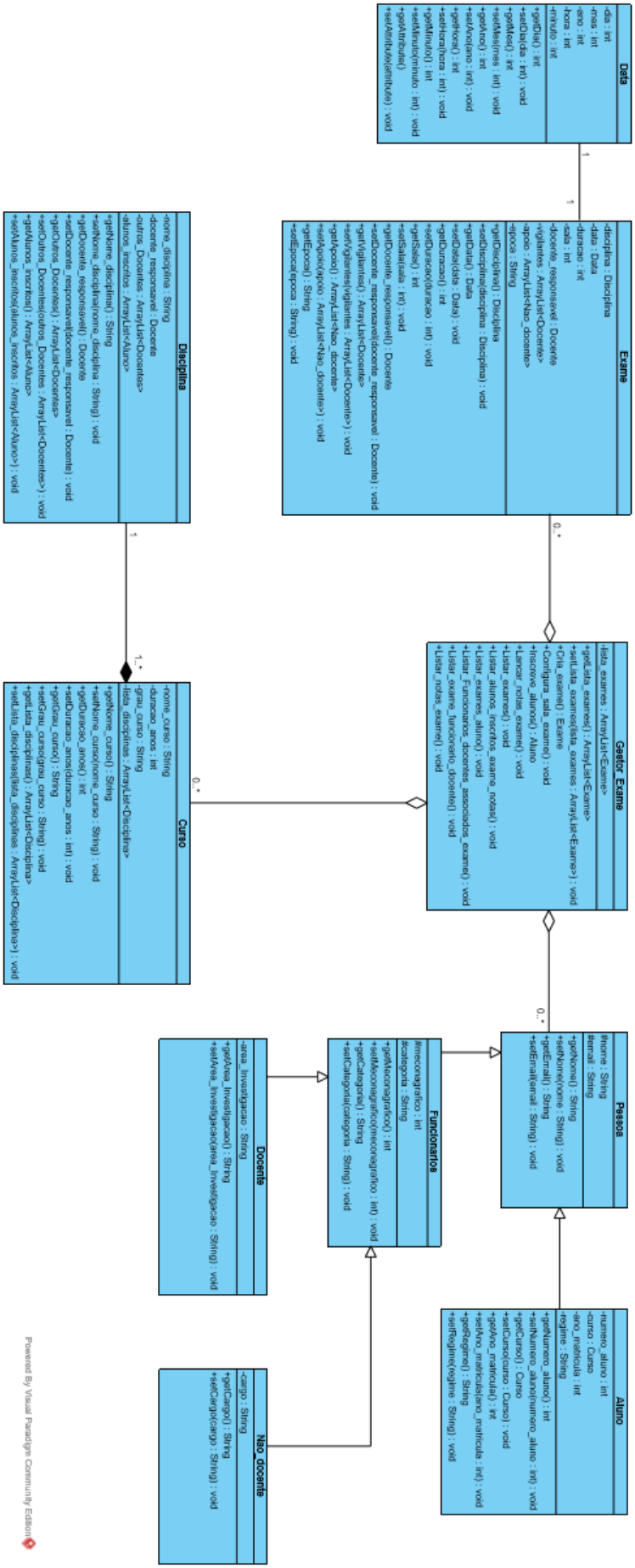
Recurso(Disciplina disciplina, int duracao, Sala sala,Docente docente_responavel, ArrayList<Docente> vigilantes, ArrayList<NaoDocentes> apoio, HashMap<Aluno, Integer> alunos, Date exame_inicio, Date exame_final);

Sala

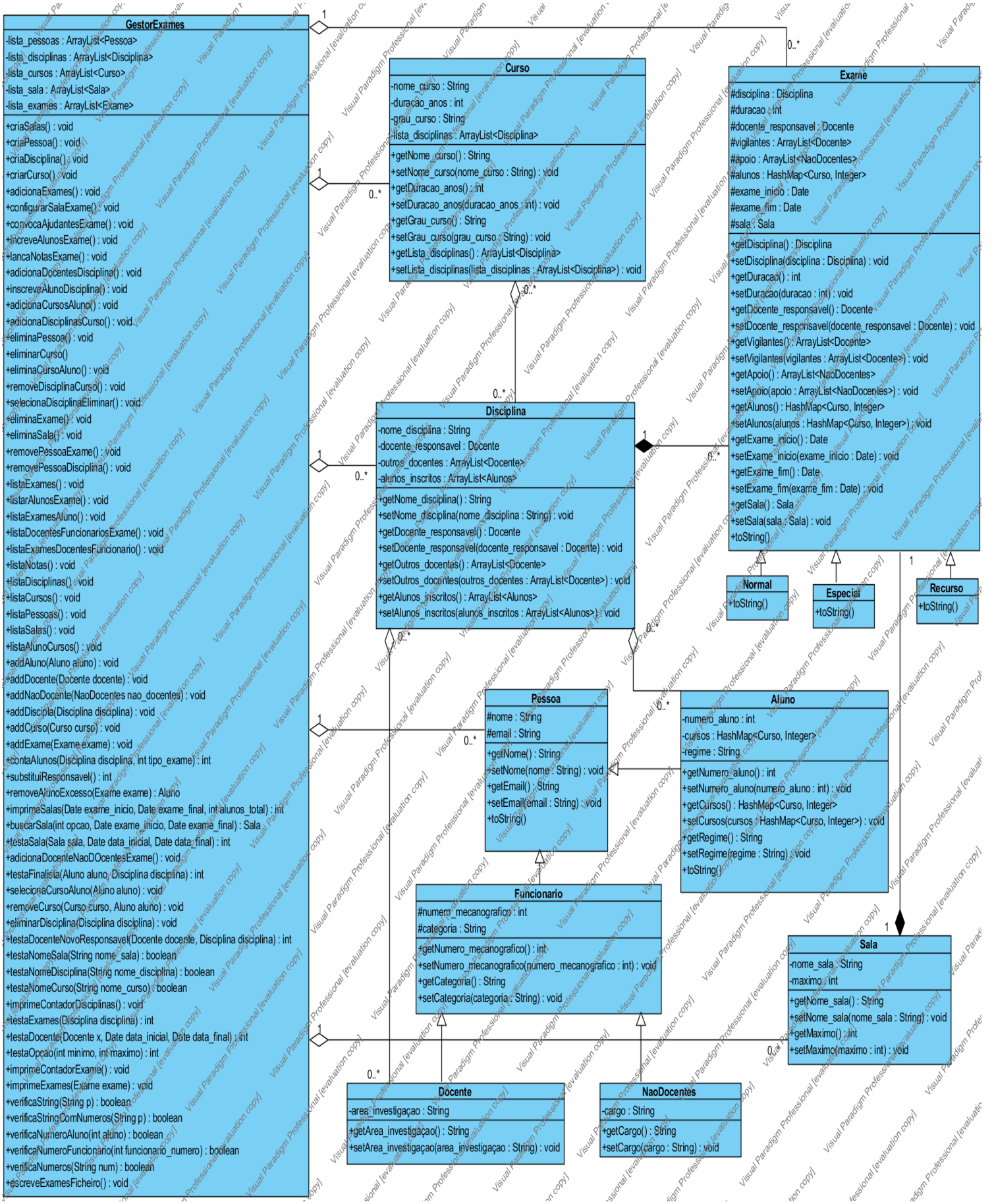
Atributos:

A classe Sala tem como atributos o nome da sala e o máximo de alunos que podem estar nela. Construtor: Sala(String nome_sala, int maximo);

1º UML


Data
-dia : int
-mes : int
-ano : int
-hora : int
-minuto : int
+getDia() : int
+setDia(dia : int) : void
+getMes() : int
+setMes(mes : int) : void
+getAno() : int
+setAno(ano : int) : void
+getHora() : int
+setHora(hora : int) : void
+getMinuto() : int
+setMinuto(minuto : int) : void
+getAtributo()
+setAtributo(atributo) : void
Exame
-disciplina : Disciplina
-data : Data
-duracao : int
-sala : int
-docente_responsavel : Docente
-vigilantes : ArrayList<Docente>
-apoio : ArrayList<Nao_docente>
-epoca : String
+getDisciplina() : Disciplina
+setDisciplina(disciplina : Disciplina) : void
+getData() : Data
+setData(data : Data) : void
+getDuracao() : int
+setDuracao(duracao : int) : void
+getSala() : int
+setSala(sala : int) : void
+getDocente_responsavel() : Docente
+setDocente_responsavel(docente_responsavel : Docente) : void
+getVigilantes() : ArrayList<Docente>
+setVigilantes(vigilantes : ArrayList<Docente>) : void
+getApoio() : ArrayList<Nao_docente>
+setApoio(apoio : ArrayList<Nao_docente>) : void
+getEpoca() : String
+setEpoca(epoca : String) : void
Gestor_Exame
-lista_exames : ArrayList<Exame>
+getLista_exames() : ArrayList<Exame>
+setLista_exames(lista_exames : ArrayList<Exame>) : void
+Cria_exame() : Exame
+Configura_sala_exame() : void
+Inscreve_aluno() : Aluno
+Lancar_notas_exame() : void
+Listar_exames() : void
+Listar_alunos_inscritos_exame_notas() : void
+Listar_exames_aluno() : void
+Listar_Funcionarios_docentes_associados_exame() : void
+Listar_exame_funcionario_docente() : void
+Listar_notas_exame() : void
Disciplina
-nome_disciplina : String
-docente_responsavel : Docente
-outros_Docentes : ArrayList<DocenteAss>
-alunos_inscritos : ArrayList<Aluno>
+getNome_disciplina() : String
+setNome_disciplina(nome_disciplina : String) : void
+getDocente_responsavel() : Docente
+setDocente_responsavel(docente_responsavel : Docente) : void
+getOutros_Docentes() : ArrayList<DocenteAss>
+setOutros_Docentes(outros_Docentes : ArrayList<DocenteAss>) : void
+getAlunos_inscritos() : ArrayList<Alunos>
+setAlunos_inscritos(alunos_inscritos : ArrayList<Aluno>) : void
Curso
-nome_curso : String
-duracao_anos : int
-grau_curso : String
-lista_disciplinas : ArrayList<Disciplina>
+getNome_curso() : String
+setNome_curso(nome_curso : String) : void
+getDuracao_anos() : int
+setDuracao_anos(duracao_anos : int) : void
+getGrau_curso() : String
+setGrau_curso(grau_curso : String) : void
+getLista_disciplinas() : ArrayList<Disciplina>
+setLista_disciplinas(lista_disciplinas : ArrayList<Disciplina>) : void
Pessoa
#nome : String
#email : String
+getNome() : String
+setNome(nome : String) : void
+getEmail() : String
+setEmail(email : String) : void
Aluno
-numero_aluno : int
-curso : Curso
-ano_matricula : int
-regime : String
+getNumero_aluno() : int
+setNumero_aluno(numero_aluno : int) : void
+getCurso() : Curso
+setCurso(curso : Curso) : void
+getAno_matricula() : int
+setAno_matricula(ano_matricula : int) : void
+getRegime() : String
+setRegime(regime : String) : void
Funcionarios
#mecanografico : int
#categoria : String
+getMecanografico() : int
+setMecanografico(mecanografico : int) : void
+getCategoria() : String
+setCategoria(categoria : String) : void
Docente
-area_Investigacao : String
+getArea_Investigacao() : String
+setArea_Investigacao(area_Investigacao : String) : void
Nao_docente
-cargo : String
+getCargo() : String
+setCargo(cargo : String) : void
1
1
0..*
1
1..*
0..*
0..*
Powered By Visual Paradigm Community Edition

2º UML


GestorExames
-lista_pessoas : ArrayList<Pessoa>
-lista_disciplinas : ArrayList<Disciplina>
-lista_cursos : ArrayList<Curso>
-lista_sala : ArrayList<Sala>
-lista_exames : ArrayList<Exame>
+criaSalas() : void
+criaPessoa() : void
+criaDisciplina() : void
+criarCurso() : void
+adicionaExames() : void
+configurarSalaExame() : void
+convocaAjudantesExame() : void
+increveAlunosExame() : void
+lancaNotasExame() : void
+adicionaDocentesDisciplina() : void
+inscreveAlunoDisciplina() : void
+adicionaCursosAluno() : void
+adicionaDisciplinasCurso() : void
+eliminaPessoa() : void
+eliminarCurso()
+eliminaCursoAluno() : void
+removeDisciplinaCurso() : void
+selecionaDisciplinaEliminar() : void
+eliminaExame() : void
+eliminaSala() : void
+removePessoaExame() : void
+removePessoaDisciplina() : void
+listaExames() : void
+listarAlunosExame() : void
+listaExamesAluno() : void
+listaDocentesFuncionariosExame() : void
+listaExamesDocentesFuncionario() : void
+listaNotas() : void
+listaDisciplinas() : void
+listaCursos() : void
+listaPessoas() : void
+listaSalas() : void
+listaAlunoCursos() : void
+addAluno(Aluno aluno) : void
+addDocente(Docente docente) : void
+addNaoDocente(NaoDocentes nao_docentes) : void
+addDiscipla(Disciplina disciplina) : void
+addCurso(Curso curso) : void
+addExame(Exame exame) : void
+contaAlunos(Disciplina disciplina, int tipo_exame) : int
+substituiResponsavel() : int
+removeAlunoExcesso(Exame exame) : Aluno
+imprimeSalas(Date exame_inicio, Date exame_final, int alunos_total) : int
+buscarSala(int opcao, Date exame_inicio, Date exame_final) : Sala
+testaSala(Sala sala, Date data_inicial, Date data_final) : int
+adicionaDoceniteNaoDocentesExame() : void
+testaFinalista(Aluno aluno, Disciplina disciplina) : int
+selecionaCursoAluno(Aluno aluno) : void
+removeCurso(Curso curso, Aluno aluno) : void
+eliminarDisciplina(Disciplina disciplina) : void
+testaDocenteNovoResponsavel(Docente docente, Disciplina disciplina) : int
+testaNomeSala(String nome_sala) : boolean
+testaNomeDisciplina(String nome_disciplina) : boolean
+testaNomeCurso(String nome_curso) : boolean
+imprimeContadorDisciplinas() : void
+testaExames(Disciplina disciplina) : int
+testaDocente(Docente x, Date data_inicial, Date data_final) : int
+testaOpcao(int minimo, int maximo) : int
+imprimeContadorExame() : void
+imprimeExames(Exame exame) : void
+verificaString(String p) : boolean
+verificaStringComNumeros(String p) : boolean
+verificaNumeroAluno(int aluno) : boolean
+verificaNumeroFuncionario(int funcionario_numero) : boolean
+verificaNumeros(String num) : boolean
+escreveExamesFicheiro() : void
Curso
-nome_curso : String
-duracao_anos : int
-grau_curso : String
-lista_disciplinas : ArrayList<Disciplina>
+getNome_curso() : String
+setNome_curso(nome_curso : String) : void
+getDuracao_anos() : int
+setDuracao_anos(duracao_anos : int) : void
+getGrau_curso() : String
+setGrau_curso(grau_curso : String) : void
+getLista_disciplinas() : ArrayList<Disciplina>
+setLista_disciplinas(lista_disciplinas : ArrayList<Disciplina>) : void
Exame
#disciplina : Disciplina
#duracao : int
#docente_responsavel : Docente
#vigilantes : ArrayList<Docente>
#apoio : ArrayList<NaoDocentes>
#alunos : HashMap<Curso, Integer>
#exame_inicio : Date
#exame_fim : Date
#sala : Sala
+getDisciplina() : Disciplina
+setDisciplina(disciplina : Disciplina) : void
+getDuracao() : int
+setDuracao(duracao : int) : void
+getDocente_responsavel() : Docente
+setDocente_responsavel(docente_responsavel : Docente) : void
+getVigilantes() : ArrayList<Docente>
+setVigilantes(vigilantes : ArrayList<Docente>) : void
+getApoio() : ArrayList<NaoDocentes>
+setApoio(apoio : ArrayList<NaoDocentes>) : void
+getAlunos() : HashMap<Curso, Integer>
+setAlunos(alunos : HashMap<Curso, Integer>) : void
+getExame_inicio() : Date
+setExame_inicio(exame_inicio : Date) : void
+getExame_fim() : Date
+setExame_fim(exame_fim : Date) : void
+getSala() : Sala
+setSala(sala : Sala) : void
+toString()
Disciplina
-nome_disciplina : String
-docente_responsavel : Docente
-outros_docentes : ArrayList<Docente>
-alunos_inscritos : ArrayList<Alunos>
+getNome_disciplina() : String
+setNome_disciplina(nome_disciplina : String) : void
+getDocente_responsavel() : Docente
+setDocente_responsavel(docente_responsavel : Docente) : void
+getOutros_docentes() : ArrayList<Docente>
+setOutros_docentes(outros_docentes : ArrayList<Docente>) : void
+getAlunos_inscritos() : ArrayList<Alunos>
+setAlunos_inscritos(alunos_inscritos : ArrayList<Alunos>) : void
Normal
+toString()
Especial
+toString()
Recurso
+toString()
Pessoa
#nome : String
#email : String
+getNome() : String
+setNome(nome : String) : void
+getEmail() : String
+setEmail(email : String) : void
+toString()
Aluno
-numero_aluno : int
-cursos : HashMap<Curso, Integer>
-regime : String
+getNumero_aluno() : int
+setNumero_aluno(numero_aluno : int) : void
+getCursos() : HashMap<Curso, Integer>
+setCursos(cursos : HashMap<Curso, Integer>) : void
+getRegime() : String
+setRegime(regime : String) : void
+toString()
Funcionario
#numero_mecanografico : int
#categoria : String
+getNumero_mecanografico() : int
+setNumero_mecanografico(numero_mecanografico : int) : void
+getCategoria() : String
+setCategoria(categoria : String) : void
Sala
-nome_sala : String
-maximo : int
+getNome_sala() : String
+setNome_sala(nome_sala : String) : void
+getMaximo() : int
+setMaximo(maximo : int) : void
Docente
-area_investigaçao : String
+getArea_investigaçao() : String
+setArea_investigaçao(area_investigaçao : String) : void
NaoDocentes
-cargo : String
+getCargo() : String
+setCargo(cargo : String) : void

Execução do programa

No início do programa o programa vai ler o ficheiro de objetos que guarda a informação ao longo da execução do programa.

Não podem ser criados exames sem salas e sem disciplinas, as disciplinas não podem ser criadas se não houver docentes responsáveis. Ao criar um exame Normal os alunos inscritos nessa disciplina são automaticamente inscritos se a sala toda tiver espaço suficiente para todos os alunos, se não só alguns serão inscritos, no caso de recurso ou especial os alunos terão de ser inscritos um a um.

Só é possível inscrever um aluno num exame se eles estiverem inscritos a essa disciplina, ele só poderá ser inscrito às disciplinas dos cursos onde esta inscrito.

No caso dos exames Especiais só poderão ser inscritos os alunos que são trabalhador-estudante, atletas, dirigente associativo ou finalistas do curso da disciplina onde pretendem realizar o exame.

Nas opções de remover se remover alguma docente responsável ele poderá ser substituído por outro caso haja mais algum, e caso não seja tanto a disciplina/s que ele era responsável como os exames dessa disciplina serão removidos.

Ao remover um curso de um aluno irá também remove-lo de todas as disciplinas ao curso associado, a não ser que essas disciplinas se encontrem noutro curso onde o aluno também esta inscrito.

Ao remover uma disciplina de um curso todos os alunos que só tinham essa disciplina através desse curso deixaram de estar inscritos nela e nos exames a ela associado.

No final de cada opção o ficheiro de objetos irá ser novamente recarregado com os novos dados, para sair do programa bastara se por 0 no menu inicial.