Universidade do Porto

Faculdade de Engenharia

FEUP

Relatório do 1º Projecto

Gestão de uma Clínica Médica

Cátia Cruz: ei08134@fe.up.pt

Gaspar Furtado: ei08072@fe.up.pt

Miao Sun: ei08162@fe.up.pt

Grupo 5 <Turma 4>

Algoritmos e Estruturas de Dados

Prof. Ana Paula Cunha da Rocha

Prof. Rosaldo José Fernandes Rossetti

Prof. Antonio Jesus Monteiro de Castro

Mestrado Integrado em Engenharia Informática e Coumputação

2010-11-03

1 – Descrição do Trabalho:

Este projecto foi desenvolvido no âmbito da disciplina de Algoritmos de Estruturas de Dados usando como plataforma de desenvolvimento as ferramentas Eclipse e Visual Studio 2010.

O objectivo deste trabalho é desenvolver uma aplicação em C++ capaz de gerir uma Clínica Médica. Pretende-se que a aplicação consiga:

- Gerir as pessoas pertencentes à clínica;
- Gerir as marcações efectuadas;

Nesta clínica médica é possível fazer uma gestão das pessoas que fazem parte da clínica, sendo elas, médicos, doentes e funcionários. Como gestão entende-se adicionar mais membros à clínica, através da adição dos seus dados pessoais, editar, caso necessário, os dados de cada um dos membros da clínica e apaga-los. O mesmo acontece com a gestão das marcações, onde é possível também adicionar, editar e apagar uma marcação.

2.2 – Implementação das classes e algoritmos utilizados

O projecto foi iniciado pela construção de todas as classes em modelo UML, seguido de uma análise para perceber como iria ser concebido na linguagem de programação C++. Por fim passou-se à implementação do código. À medida que ia sendo investido tempo no projecto e consoante o aparecimento de novas ideias, o modelo UML sofreu algumas alterações.

Neste projecto optou-se pela implementação de oito classes (Manutenção, Marcação, Exame, Consulta, Pessoa, Médico, Paciente e Funcionário).

- Classe Pessoa: Classe base das classes Médico, Paciente e Funcionário. É uma classe puramente virtual, ou seja, nenhum objecto se cria a partir desta classe mas sim das suas subclasses. Contém como atributos o nome, data de nascimento, o ID (único de cada Pessoa), e tipo (que define qual o tipo de Pessoa: Médico, Funcionário ou Doente).

- Classe Médico: Classe derivada da classe Pessoa. Para além de herdar os atributos derivados de Pessoa acrescenta também os atributos Especialidade, Horário e Vencimento à classe.

- Classe Doente: Classe derivada da classe Pessoa. Para além dos atributos derivados de Pessoa acrescenta também os atributos Morada à classe.

- Classe Funcionário: Classe derivada da classe Pessoa. Para além dos atributos de Pessoa acrescenta também os atributos Cargo e Vencimento à classe.

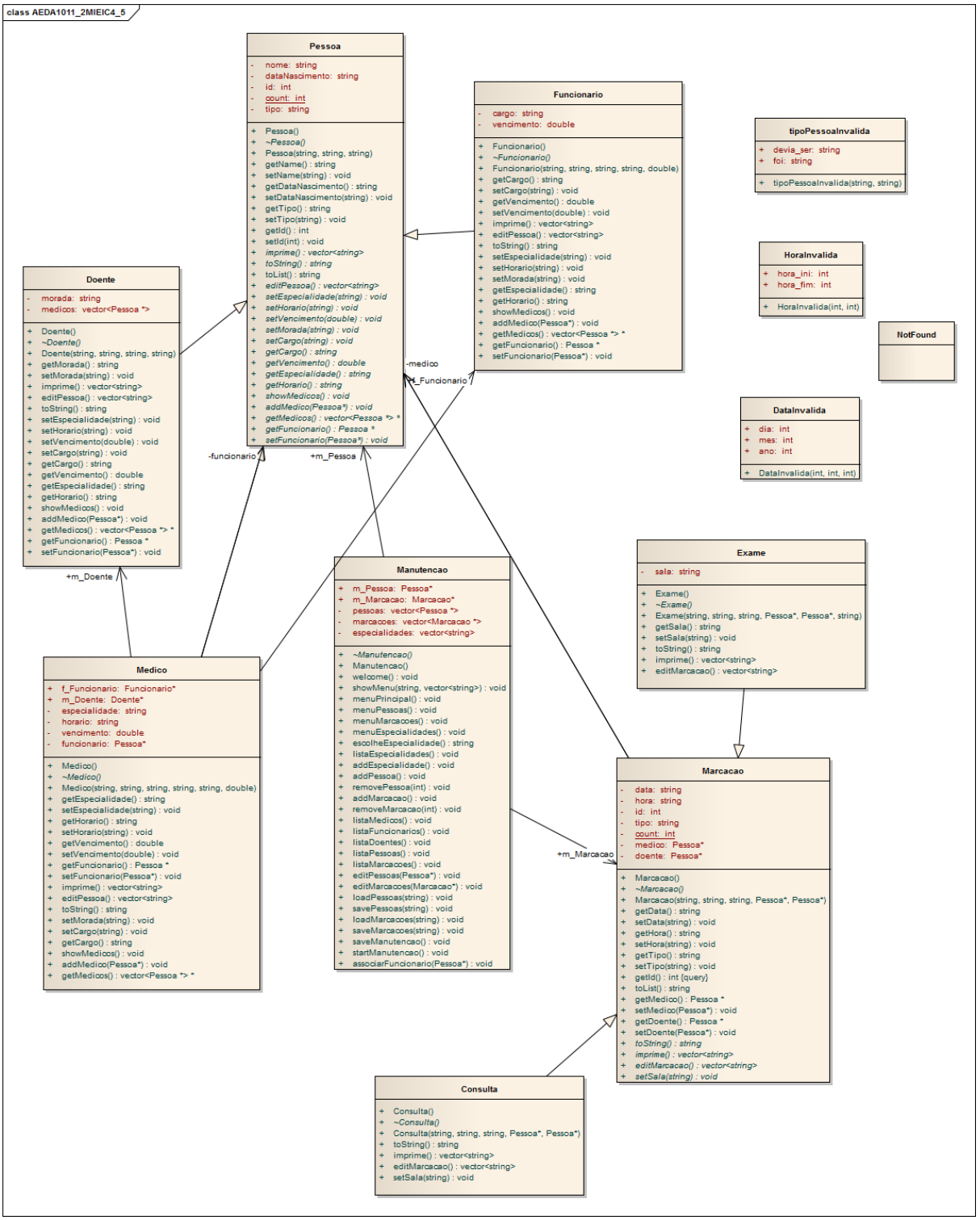
- Classe Marcação: Classe base das classes Consulta e Exame. É puramente virtual, ou seja, os objectos são criados a partir das suas classes derivadas. Tem como atributos data, hora, ID (único para cada Marcação) e tipo (que define qual o tipo de Marcação: Consulta ou Exame)

-Classe Consulta: Classe derivada da classe Marcação. Os seus atributos derivam da classe Marcação.

-Classe Exame: Classe derivada da classe Marcação. Para além dos atributos derivados de Pessoa acrescenta também o atributo sala.

-Classe Manutenção: Classe Principal. Classe onde se dá a gestão de Pessoas e de Marcações.

3 – UML:


class AEDA1011_2MIEIC4_5
Pessoa
- nome: string
- dataNascimento: string
- id: int
- count: int
- tipo: string
+ Pessoa()
+ ~Pessoa()
+ Pessoa(string, string, string)
+ getName() : string
+ setName(string) : void
+ getDataNascimento() : string
+ setDataNascimento(string) : void
+ getTipo() : string
+ setTipo(string) : void
+ getId() : int
+ setId(int) : void
+ imprime() : vector<string>
+ toString() : string
+ toList() : string
+ editPessoa() : vector<string>
+ setEspecialidade(string) : void
+ setHorario(string) : void
+ setVencimento(double) : void
+ setMorada(string) : void
+ setCargo(string) : void
+ getCargo() : string
+ getVencimento() : double
+ getEspecialidade() : string
+ getHorario() : string
+ showMedicos() : void
+ addMedico(Pessoa*) : void
+ getMedicos() : vector<Pessoa *> *
+ getFuncionario() : Pessoa *
+ setFuncionario(Pessoa*) : void
Funcionario
- cargo: string
- vencimento: double
+ Funcionario()
+ ~Funcionario()
+ Funcionario(string, string, string, string, double)
+ getCargo() : string
+ setCargo(string) : void
+ getVencimento() : double
+ setVencimento(double) : void
+ imprime() : vector<string>
+ editPessoa() : vector<string>
+ toString() : string
+ setEspecialidade(string) : void
+ setHorario(string) : void
+ setMorada(string) : void
+ getEspecialidade() : string
+ getHorario() : string
+ showMedicos() : void
+ addMedico(Pessoa*) : void
+ getMedicos() : vector<Pessoa *> *
+ getFuncionario() : Pessoa *
+ setFuncionario(Pessoa*) : void
tipoPessoaInvalida
+ devia_ser: string
+ foi: string
+ tipoPessoaInvalida(string, string)
HoraInvalida
+ hora_ini: int
+ hora_fim: int
+ HoraInvalida(int, int)
NotFound
DataInvalida
+ dia: int
+ mes: int
+ ano: int
+ DataInvalida(int, int, int)
Doente
- morada: string
- medicos: vector<Pessoa *>
+ Doente()
+ ~Doente()
+ Doente(string, string, string, string)
+ getMorada() : string
+ setMorada(string) : void
+ imprime() : vector<string>
+ editPessoa() : vector<string>
+ toString() : string
+ setEspecialidade(string) : void
+ setHorario(string) : void
+ setVencimento(double) : void
+ setCargo(string) : void
+ getCargo() : string
+ getVencimento() : double
+ getEspecialidade() : string
+ getHorario() : string
+ showMedicos() : void
+ addMedico(Pessoa*) : void
+ getMedicos() : vector<Pessoa *> *
+ getFuncionario() : Pessoa *
+ setFuncionario(Pessoa*) : void
Exame
- sala: string
+ Exame()
+ ~Exame()
+ Exame(string, string, string, Pessoa*, Pessoa*, string)
+ getSala() : string
+ setSala(string) : void
+ toString() : string
+ imprime() : vector<string>
+ editMarcacao() : vector<string>
Manutencao
+ m_Pessoa: Pessoa*
+ m_Marcacao: Marcacao*
- pessoas: vector<Pessoa *>
- marcacoes: vector<Marcacao *>
- especialidades: vector<string>
+ ~Manutencao()
+ Manutencao()
+ welcome() : void
+ showMenu(string, vector<string>) : void
+ menuPrincipal() : void
+ menuPessoas() : void
+ menuMarcacoes() : void
+ menuEspecialidades() : void
+ escolheEspecialidade() : string
+ listaEspecialidades() : void
+ addEspecialidade() : void
+ addPessoa() : void
+ removePessoa(int) : void
+ addMarcacao() : void
+ removeMarcacao(int) : void
+ listaMedicos() : void
+ listaFuncionarios() : void
+ listaDoentes() : void
+ listaPessoas() : void
+ listaMarcacoes() : void
+ editPessoas(Pessoa*) : void
+ editMarcacoes(Marcacao*) : void
+ loadPessoas(string) : void
+ savePessoas(string) : void
+ loadMarcacoes(string) : void
+ saveMarcacoes(string) : void
+ saveManutencao() : void
+ startManutencao() : void
+ associarFuncionario(Pessoa*) : void
Medico
+ f_Funcionario: Funcionario*
+ m_Doente: Doente*
- especialidade: string
- horario: string
- vencimento: double
- funcionario: Pessoa*
+ Medico()
+ ~Medico()
+ Medico(string, string, string, string, string, double)
+ getEspecialidade() : string
+ setEspecialidade(string) : void
+ getHorario() : string
+ setHorario(string) : void
+ getVencimento() : double
+ setVencimento(double) : void
+ getFuncionario() : Pessoa *
+ setFuncionario(Pessoa*) : void
+ imprime() : vector<string>
+ editPessoa() : vector<string>
+ toString() : string
+ setMorada(string) : void
+ setCargo(string) : void
+ getCargo() : string
+ showMedicos() : void
+ addMedico(Pessoa*) : void
+ getMedicos() : vector<Pessoa *> *
Marcacao
- data: string
- hora: string
- id: int
- tipo: string
- count: int
- medico: Pessoa*
- doente: Pessoa*
+ Marcacao()
+ ~Marcacao()
+ Marcacao(string, string, string, Pessoa*, Pessoa*)
+ getData() : string
+ setData(string) : void
+ getHora() : string
+ setHora(string) : void
+ getTipo() : string
+ setTipo(string) : void
+ getId() : int {query}
+ toList() : string
+ getMedico() : Pessoa *
+ setMedico(Pessoa*) : void
+ getDoente() : Pessoa *
+ setDoente(Pessoa*) : void
+ toString() : string
+ imprime() : vector<string>
+ editMarcacao() : vector<string>
+ setSala(string) : void
Consulta
+ Consulta()
+ ~Consulta()
+ Consulta(string, string, string, Pessoa*, Pessoa*)
+ toString() : string
+ imprime() : vector<string>
+ editMarcacao() : vector<string>
+ setSala(string) : void
-medico
+m_Funcionario
-funcionario
+m_Pessoa
+m_Doente
+m_Marcacao

4 – Casos de Utilização da aplicação:

O gestor da base de dados pode:

- Adicionar uma Pessoa do tipo Médico, Doente ou Funcionário ao sistema;
- Adicionar uma Marcação do tipo Consulta ou Exame ao sistema;
- Editar qualquer um dos atributos de um Médico, Doente ou Funcionário;
- Fazer alterações às Marcações feitas anteriormente;
- Atribuir um funcionário a cada médico;
- Atribuir um conjunto de médicos a cada doente, que o acompanham;
- Apagar qualquer Pessoa registada no sistema;
- Apagar qualquer Marcação;

Relato das principais dificuldades encontradas no desenvolvimento do trabalho:

Para armazenamento das pessoas ligadas ao sistema optou-se por uma única estrutura linear de dados (vector) da superclasse Pessoa, e, em consequência disso, a primeira dificuldade foi a de conseguir distinguir entre os três tipos de pessoas. Para tal, a solução que se arranjou foi ter um atributo "tipo" na classe Pessoa que identifica o tipo de pessoa que se trata. Como consequência deste tipo de armazenamento de dados, houve a necessidade de criar um conjunto elevado de funções abstractas, específicas de cada uma das subclasses.