

Algoritmos e Estruturas de Dados

Gestão de uma Clínica Médica

Cátia Cruz – 080509134 - [ei08134@fe.up.pt](mailto:ei08134@fe.up.pt)

Gaspar Furtado – 080509072 – [ei08072@fe.up.pt](mailto:ei08072@fe.up.pt)

Miao Sun – 080509162 – [ei08162@fe.up.pt](mailto:ei08162@fe.up.pt)

03 de Novembro de 2010

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. Implementação:

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

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.