 UNIVERSIDADE DE COIMBRA FACULDADE DE CIÊNCIAS E TECNOLOGIA *Departamento de Engenharia Informática*	Programação Orientada a Objectos / Programação e Algoritmos III Projecto 2008/09 SISTEMA DE GESTÃO DE PEDIDOS DE SUPORTE TÉCNICO

Manual do Programador

André Fernandes de Carvalho
2006130876 - LEI
afcarv@student.dei.uc.pt

Daniel Ferreira
2007103439 – LEI
dfof@student.dei.uc.pt

Índice

(a) Estrutura geral do programa;

O objectivo deste trabalho prático consistia em criar em linguagem java, um sistema de gestão de pedidos de suporte técnico informático de uma organização.
Cada utilizador pode fazer um pedido de suporte relativo ao sistema (Hardware/Software) que utiliza ou relativo a formação sobre um determinado assunto. Existem, portanto, pedidos de assistência de 'Sistema' ou de 'Formação'.
O registo de um pedido de suporte deve conter a seguinte informação: N.º do pedido, N.º do Utilizador e Data. Caso se trate de um pedido de suporte de Sistema, deve ainda incluir o N.º do posto de trabalho e a descrição do problema; caso se trate de um pedido de Formação, deve conter uma indicação do Tema.
A resolução de um pedido de suporte pode passar por várias intervenções, por parte de um ou mais técnicos (um técnico por intervenção). Cada intervenção é composta por um N.º de intervenção, N.º do pedido de suporte a que se refere, N.º do Técnico que fez a intervenção, descrição, Data, Duração e se ficou ou não resolvido.
Para cada posto de trabalho, devem ser registadas as seguintes informações: N.º, CPU, RAM, Disco e Sistema Operativo.
O sistema a desenvolver deve, ainda, manter os registos quer dos Técnicos, quer dos Utilizadores. Sobre cada um, é registado o N.º e o Nome. Deverá existir uma numeração distinta para Técnicos e para Utilizadores.
Este sistema deverá permitir fazer a manutenção (criar, alterar, eliminar e listar) de registos de Utilizadores, Técnicos e Postos de Trabalho e ainda, permitir fazer a manutenção (criar, alterar, eliminar e listar) de Pedidos de Assistência e de Intervenções dos pedidos ainda não resolvidos.
Informações como estatísticas para um determinado período de tempo, e para cada tipo de pedido (Sistema ou Formação) são também necessárias, como por exemplo:
• N.º de pedidos;
• N.º de pedidos resolvidos;
• Tempo médio de resolução (ter em conta que cada pedido pode ter várias intervenções até à sua resolução);
• N.º médio de intervenções por pedido resolvido.

A informação gerida por este sistema será mantida nos seguintes ficheiros:

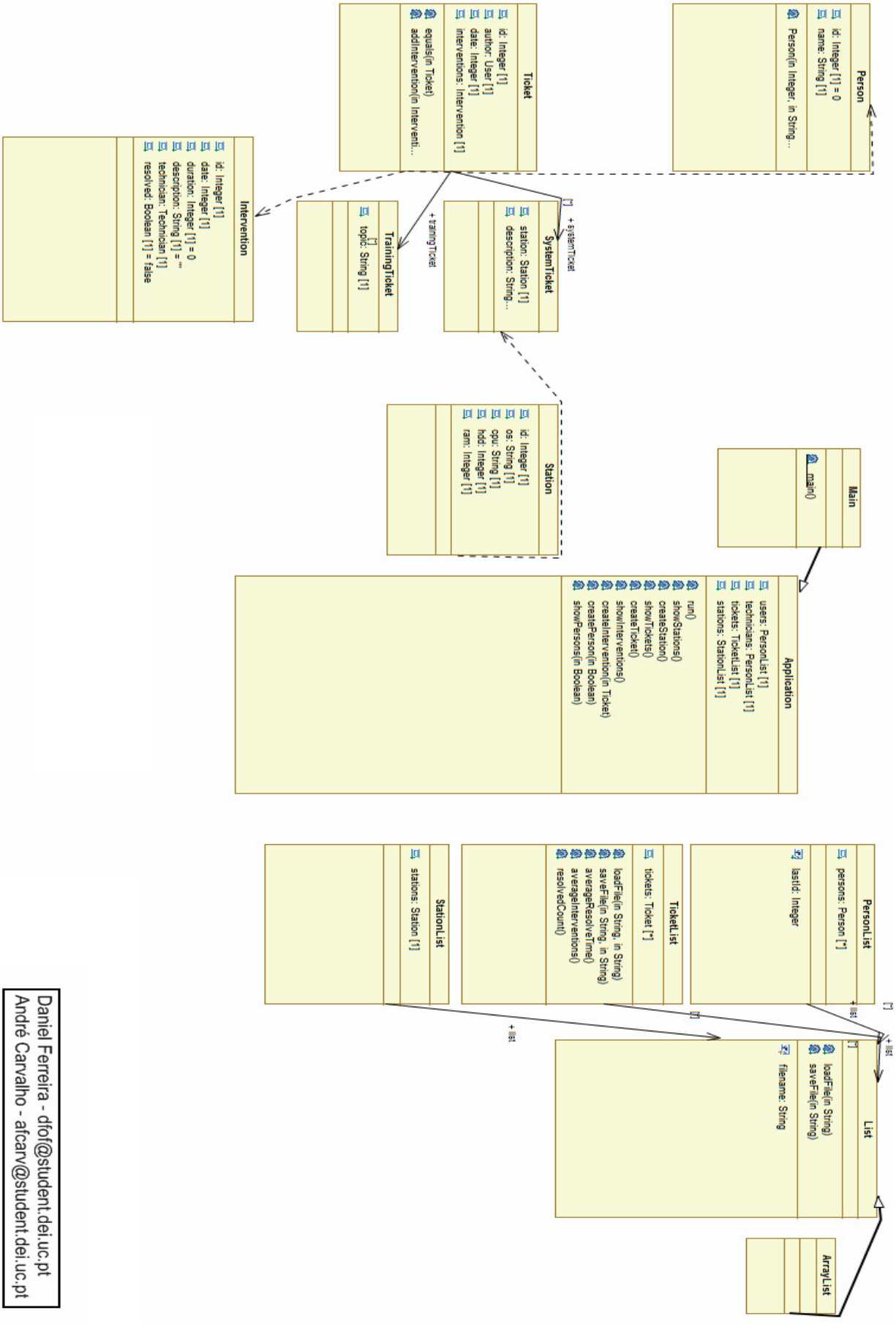
Registos	Ficheiro	Tipo de Ficheiro
Pessoas	Pessoas.dat	Binário
Postos de Trabalho	Postos_trab.dat	Binário
Pedidos de Assistência	Pedidos.txt	Texto
Intervenções	Intervencoes.txt	Texto

Alguns dos requisitos para a implementação desta aplicação são:

1. Cada classe deve gerir internamente os seus dados, pelo que deverá cuidar da protecção das suas variáveis e métodos;
2. Cada objecto deverá ser responsável por uma tarefa ou objectivo específico, não lhe devendo ser atribuídas funções indevidas;
3. A keyword *static* apenas deverá ser usada quando tal se justifique e não para contornar erros do compilador.

b. Diagramas de classes:

Esta é a versão final do Diagrama de classes para este projecto:



Daniel Ferreira - dfof@student.dei.uc.pt
André Carvalho - afcarv@student.dei.uc.pt

Esta versão do diagrama de classes foi fruto de uma conversa com o docente das teórico-práticas, aquando da avaliação da primeira meta.
A primeira versão do diagrama segue em baixo:

c. Descrição das principais estruturas de dados e de ficheiros usados

Para a realização deste projecto foram concebidas as seguintes doze classes:

Main.java n2007103439_TicketList.java n2006130876_Intervention.java

n2006130876_Person.java n2006130876_Station.java

n2006130876_SystemTicket.java n2006130876_Ticket.java

n2006130876_TrainningTicket.java n2007103439_Application.java

n2007103439_CustomList.java n2007103439_StationList.java

Segue uma explicação de cada uma destas classes.

Main – Esta classe serve somente para iniciar uma nova aplicação, criando um novo objecto da classe Application.

```
application app = new Application();
app.run();
```

Application – Esta classe é responsável pela administração do fluxo do programa. Esta é a classe responsável pela interacção com o utilizador, lidando com esses mesmos dados, interagindo com as outras classes necessárias ao desenrolar do programa, ou seja, contém a interface do sistema.
Quanto aos menus são três, um showMainMenu que modela a interacção primária do utilizador com o sistema, um showManageOthersMenu que é responsável pela interacção ao nível de gestão de dados e por último o showEditOptions que trata a interacção do utilizador com o sistema em termos de edição a alteração de dados.

CustomList – Esta classe contém a lista responsável pelo armazenamento de objectos armazenáveis sejam eles do tipo person, ticket ou station. Faz extend a uma array list para guardar os elementos necessários.
Alguns dos atributos definidos são filename, usado para carregar ou guarder dados, lastid onde é guardado o último id e que é devolvido com o auxílio da função getLastid().

Outra das funções presentes nesta classe é a getById(int id), que devolve um objecto dado um determinado id, à semelhança do que o método findid(id) faz ou seja, dado um id, determina a posição desse elemento.
Passando agora à função do tipo boolean, add. Esta função adiciona um elemento à lista, incrementando o lastId, para fazer overide do id do objecto a ser armazenado.
Quanto à função, também do tipo Boolean, delete, é responsável pelo eliminar de um elemento dado o seu id.
Passando agora à função de leitura, readCount, esta carrega uma lista de um ficheiro e com o auxílio do filename, que usa como target, carrega o ficheiro. Depois disto procede à leitura do mesmo.
Quanto à função writeCount, funciona de modo semelhante mas em vez de leitura procede à escrita de um ficheiro.
clearFile é a função responsável por "limpar" o conteúdo de um ficheiro, abrindo-o para escrita e depois sobrepondo o seu conteúdo com conteúdo vazio.
As funções clPrintAll, clPrint, são funções para imprimir o conteúdo do ficheiro.
clNew serve para adicionar um novo elemento à lista.
clEdit é responsável pelo update de um elemento.
clDelete serve para elminar um dado elemnto da lista.

personList – Esta classe serve para guarder objectos do tipo person, daí que seja um extend da classe CustomList para objectos modelados na classe, abordada mais tarde, person.
Para além dos atributos próprios da classe person, type, esta classe possui as funções add, getById, findId, loadFile, saveFile, clPrintAll, clPrint, clNew, clEdit necessárias para edição de elementos na lista do tipo person sendo as funções responsáveis pela escrita no ficheiro de alterações como adicionar ou eliminar elementos ou carregar do ficheiro elementos.

TicketList – Esta lista, ligada à customList, é um extend, serve para guardar objectos do tipo Ticket, da classe ticket que será falada mais tarde.
O ficheiro interventionFile serve para guardar as intervenções efectuadas.
StationList posts é o ponteiro para a lista das Stations.
PersonList techs é o ponteiro para a lista de técnicos.
Por fim, PersonList users é o ponteiro para a lista de utilizadores.
Date averageResolveTime serve para devolver a média de tempo para todos os tickets, a partir da data desejada.
averageInterventions devolve o número de intervenções para todos os bilhetes.
A função loadFile serve para carregar um determinado ficheiro e de uma forma semelhante, a função saveFile serve para guardar num ficheiro o conteúdo desejado.
A função clPrintAll imprime os bilhetes e respectiva informação adjacente.
A função clNew, do tipo boolean serve para criar um novo ticket.

StationList – Esta classe, guarda objectos do tipo Station, à semelhança da classe TicketList em relação aos objectos ticket e possui igualmente funções para adicionar uma nova station (add()), eliminar (delete()), pesquisar (getById() e findId()), carregar um ficheiro ou guardar as alterações do mesmo (loadFile() e saveFile()), sendo que os mecanismos de detecção de erros foram os mesmos que se implementaram em TicketList. Ainda referência à função clEdit(id), que permite alterar os dados de uma estação.

Ticket – Esta classe serve para modelar um ticket ou senha.
As características de um ticket são, type, que define o tipo de senha, ticket id um valor numérico único para um determinado ticket, date que representa a data do ticket, person author representa a pessoa que requisitou a senha, o arrayLisyt interventions contém a lista de todas as intervenções.
O método toString, devolve a informação do ticket.
O método isResolved, do tipo boolean, serve para verificar se o ticket foi tratado.
O método timesResolveddevolve o número de vezes que uma senha foi marcada como resolvida.
O método resolveTime devolve o tempo que uma senha demorou a ser tratada, em milissegundos.

SystemTicket – Esta classe representa a senha de sistema e é um extend da classe ticket. Nesta classe é definida a estação para a qual é requisitada a senha e a descrição do problema.

TrainningTicket – Esta classe representa um pedido de ticket, pedido de formação e é também uma extensão da classe Ticket. Um dos seus atributos é o tópico e tem ainda um construtor com o autor, a data e o tópico.

Person – Esta classe modela e representa uma pessoa. Cada pessoa possui um id único. Esta classe como atributos tem as características para uma pessoa. Ainda de referir que a pessoa pode ser um utilizador ou um técnico, user ou worker, e para a distinção atribuímos valores específicos a cada tipo:

```
public static final int USER = 1;
 public static final int WORKER = 2;
```

Station – Esta classe é responsável pela definição dos atributos de armazenamento da informação de uma estação. Alguns dos atributos desta classe são sistema operativo, cpu, hdd e ram. Também está definido um construtor da classe, com os argumentos os, cpu, hdd e ram.

Intervention – Por ultimo a classe que representa uma intervenção. Como atributos esta classe tem o id da intervenção, a duração, a descrição, o técnico e um boolean que define se a senha foi tratada com sucesso ou não.

d. Javadoc
e. Listagem do programa