**AMIGO OCULTO**

**AUTOAVALIAÇÃO**

**ALUNOS:** *Rafael Mourão Cerqueira Figueiredo*

*Lucas Soriano de Oliveira*

*Luisa Mesquita Oliveira Ribeiro*

**TURNO:** *manhã*

* **Situação:** Informe as letras C (completa), I (incompleta ou com erros) ou N (não implementada) para o tanto que a funcionalidade especificada na primeira coluna foi implementada.
* **Método:** Informe o nome do método responsável pela funcionalidade especificada na primeira coluna. Não considere o método responsável pelo menu, mas aquele que executa as instruções desta funcionalidade.
* **Arquivo:** Informe o nome do arquivo .java em que o método da coluna anterior está implementado.
* **Linha:** Informe a linha desse arquivo em que o método se inicia.

**RELATÓRIO DAS IMPLEMENTAÇÕES:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **FUNCIONALIDADE** | **SITUAÇÃO** | **MÉTODO** | **ARQUIVO** | **LINHA** |
| CRUD genérico | C | Classe CRUD | CRUD.java | 7 |
| **USUÁRIOS** | | | | |
| Inclusão de novo usuário | C | novoUsuario() | Rotinas.java | 33 |
| Acesso ao sistema por email e senha | C | login() | Rotinas.java | 14 |
| **SUGESTÕES** | | | | |
| Inclusão de sugestões | C | incluirSug() | Rotinas.java | 122 |
| Listagem de sugestões | C | listarSug() | Rotinas.java | 113 |
| Alteração de sugestões | C | alterarSug() | Rotinas.java | 147 |
| Exclusão de sugestões | C | excluirSug() | Rotinas.java | 197 |
| **GRUPOS** | | | | |
| Inclusão de grupos | C | incluirGrup() | Rotinas.java | 396 |
| Listagem de grupos | C | listarGrup() | Rotinas.java | 387 |
| Alteração de grupos | C | alterarGrup() | Rotinas.java | 430 |
| Desativação de grupos | C | desativarGrup() | Rotinas.java | 484 |
| **CONVITES** | | | | |
| Emissão de convites | C | emitir() | Rotinas.java | 527 |
| Listagem de convites | C | listarConv() | Rotinas.java | 511 |
| Cancelamento de convites | C | cancelar() | Rotinas.java | 584 |
| **INSCRIÇÕES** | | | | |
| Aceitação de convites no menu inicial | C | convitesPendentes() | Rotinas.java | 56 |
| Recusa de convites no menu inicial | C | convitesPendentes() | Rotinas.java | 56 |
| **PARTICIPANTES** | | | | |
| Listagem de participantes | C | listarPart() | Rotinas.java | 627 |
| Remoção de participantes | C | removerPart() | Rotinas.java | 642 |
| Transferência do amigo do usuário removido | C | removerPart() | Rotinas.java | 642 |
| **PARTICIPAÇÕES** | | | | |
| Listagem de participantes | C | verParticipantes(int idGrupo) | Rotinas.java | 298 |
| Visualização do amigo sorteado | C | meuAmigoOculto(int idAmigo) | Rotinas.java | 306 |
| Visualização das sugestões do amigo | C | meuAmigoOculto(int idAmigo) | Rotinas.java | 306 |
| **MENSAGENS** | | | | |
| Envio de mensagens para o grupo | C | novaMensagem(int idGrupo) | Rotinas.java | 330 |
| Leitura de mensagens do grupo | C | mensagensDoGrupo(int idGrupo) | Rotinas.java | 348 |
| (+ Desafio: responder mensagens) | C | responderMensagem(int idMensagem) | Rotinas.java | 367 |
| **SORTEIO** | | | | |
| Realização do sorteio | C | sorteio() | Rotinas.java | 224 |

**OBSERVAÇÕES:**

Kutova, você vai perceber que os métodos listados na tabela não executam toda a tarefa de forma independente. Isso pois, ao fazer o projeto, percebi que as operações possuem sub-tarefas idênticas (por exemplo: o método de remoção de determinada entidade nada mais é que o processo de listagem das respectivas entidades, porém com a opção de selecionar uma delas).

Por isso, elaborei uma arquitetura com visando a economia de linhas de código e generalização de tarefas. Desta forma, se eu quisesse modificar a UI de alguma forma, não teria que sair modificando linha a linha. Além disso, elaborei a arquitetura com o objetivo de, também, simplificar minha vida para as próximas implementações que viessem a ser feitas.

Eu entendo que, talvez, minha arquitetura esteja um pouco difícil de entender à primeira vista. Peço desculpas por isso, mas eu não tenho a mínima noção de arquitetura de software, e essa foi a primeira vez que eu “brinquei” com isso. No entanto, ela é mais simples do que aparenta:

* **ESTRUTURA DE ARQUIVOS-FONTE:** Aqui, vou te explicar, mais ou menos, o que cada arquivo fonte contém (ignorando, claro, os arquivos das Árvores B+ e da HashExtensível, disponibilizadas por você).
  + **TUI\_Element.java:** TUI\_Element é uma interface, da qual duas classes implementam:
    - Rotina
    - Menu

Basicamente, a UI é composta de OPÇÕES, onde cada opção pode nos levar à execução de uma **rotina** ou à execução de um novo **menu** (lembrando que todo menu é composto de objetos de **Opção** (da classe explicada abaixo).

* + **Opção.java:** é a classe que manipula os objetos das classes que implementam a interface TUI\_Element listadas acima (da mesma forma que a classe CRUD manipula os objetos das classes que implementam a interface Entidade).
  + **TUI.java:** TUI (*Text User Interface*) é uma classe abstrata, cuja função é armazenar métodos relacionados com a UI (mensagens de erro, manipulação visual da UI, etc...).  
     Infelizmente, eu deixei os **métodos de generalização de tarefas** nesta classe (linha 97 pra baixo). Eu pretendia movê-los para a classe Rotinas (arquivo Rotinas.java), mas acabei esquecendo. O que você precisa saber é que estes métodos (a partir da linha 97) são usados somente na classe Rotinas.  
     Vale lembrar que te enviei uma mensagem no canvas explicando quase tudo que você precisa saber à respeito destes métodos de generalização de tarefas; então se você não entendê-los, da uma olhada lá.
  + **AmigoOculto.java:** corresponde à classe *Main*; inicializa os Menus e armazena alguns valores/objetos globais:
    - Todos os CRUDs
    - Todas as Árvores de relacionamento (ArvoreBMais\_Int\_Int)
    - ID do usuário logado
    - Versão do sistema (1.0)
    - String formatação de data usada no sistema
    - DateFormatter para formatação de datas.
  + **Validação.java:** classe abstrata, simplesmente armazena métodos para validação de dados (usados para ler entradas dos usuários, geralmente quando estamos criando um novo registro).
  + **Solicitação.java:** objetos de Solicitação são usados quando o sistema vai solicitar ao usuário dados genéricos (ou seja, não apenas um número, como no caso dos menus). Os atributos de uma solicitação são:
    - String mensagemErro: mensagem de erro, caso o dado inserido seja inválido.
    - String solicitacao: **o que** será solicitado. Por exemplo, “Email”.
    - Method validacao: método de validação da entrada do usuário (este método está na classe Validacao, explicada acima).
    - boolean acceptEmptyLine: dita se o valor vazio é válido para a solicitação em questão (por exemplo, ao criar um novo usuário, nenhum valor pode ser vazio).
  + **Rotinas.java:** a classe Rotinas é onde foram implementados todas as rotinas do sistema (rotinas de inclusão, alteração, exclusão, etc…). Por isso, como você pode perceber, tirando no CRUD, todos os campos de “Arquivo” na tabela deste documento são o mesmo: “Rotinas.java”.

Acredito que isso seja o suficiente para que você entenda o código sem mais problemas, Kutova. Se ainda sobrarem dúvidas, posso te ligar ou te enviar algum áudio, explicando cada detalhe (não o fiz neste documento por quê não queira deixá-lo muito grande, além de que o próprio código possúi comentários tentando explicar). Basta me avisar!