|  |  |
| --- | --- |
| **UFRGS**  **Técnicas De Construção De Programas 2015/2**  **Trabalho Prático – Genius/Simon**  **Prof. Marcelo Soares Pimenta** | **00229753 - Caroline Knorr Carvalho**  **00231060 - Guilherme de Oliveira Tassinari** |

Trabalho prático da disciplina

Genius - <https://github.com/guitassinari/Genius>

Resumo:

O projeto do jogo Genius foi feito baseando-se no *Design Pattern* MVC (Model-View-Controller), de forma que o código seja estruturado e divido em classes que trabalham exclusivamente com o que será apresentado ao usuário(View : Janela e Telas), controladores que fazem a interação com a View e manipulam os modelos(Controllers) e os modelos em si.

Foi dada grande ênfase à clareza do código, de forma que qualquer um pudesse ler e entender a essência do que está sendo realizado, através de variáveis e métodos nomeados referenciando seus papéis dentro do software, assim como uma documentação rica, clara e objetiva.

Também foi priorizada uma interface simples e intuitiva, com modais informativos aparecendo ao usuário sempre que necessário.

Requisitos:

* Funcionais:
  + Mostrar menu ao usuário para navegar entre as funcionalidades.
  + Manipular, guardar e exibir as X melhores pontuações alcançadas por usuários, onde X pode ser definido no código do projeto.
  + Receber, manipular e guardar corretamente os dados de uma partida, como nome do jogador e pontuação, para construção de ranking.
  + Responder às interações com o usuário e fornecer informações úteis sempre que necessário, incluindo efeitos sonoros e visuais.
  + Exibir tela com instruções básicas de utilização do jogo.
  + Realizar corretamente as funcionalidades da partida, como recebimento de “cliques” do usuário, geração randômica das sequencias, testes de vitória e derrota do usuário.

Projeto Funcional:*Estão listadas aquelas que fizeram parte da ideia inicial do projeto. Outras classes foram adicionadas por questões técnicas da linguagem e serão explicadas mais adiante, outras foram removidas por serem desnecessárias.*

* JanelaDeJogo:
  + Responsável pelas manipulações gerais do viewport do jogo, bem como o fluxo de telas;
* Tela:
  + São as interfaces do Jogo. O conteúdo propriamente dito que será exibido no viewport. Servirá como classe abstrata para as diversas telas do jogo, implementando métodos comuns à todas as “telas”, que serão responsáveis exclusivamente pelas interações feitas diretamente com o usuário.
* TelaDePartida:
  + Responsável por toda e qualquer interação feita com o usuário durante a partida. Emissão de efeitos sonoros, efeitos visuais, mensagens e etc.
* ControladorPartida:
  + Classe responsável pela manipulação de modelos e interação com a TelaDePartida, de forma a se complementarem. Toda a lógica da partida será feita neste controlador.
* TelaMenuPrincipal:
  + Interface gráfica que exibe as opções de “telas”.
* TelaRanking:
  + Interface gráfica que exibe as X melhores pontuações do jogo.
* ControladorRanking:
  + Classe responsável pela leitura e escrita do ranking, bem como a interação com a TelaRanking.
* TelaHelp:
  + Tela estática dando instruções básicas de utilização do programa.
* TelaInserirNome:
  + Interface gráfica onde o jogador poderá inserir seu nome para posterior gravação no ranking.
* Mensagem:
  + Classe contendo diversas Strings que serão utilizadas na interface do usuário.
* Constante:
  + Constantes numéricas que serão utilizadas no código, como o número máximo de partidas a serem exibidas no ranking.
* Cor:
  + Classe redefinindo algumas constantes de cores com novos nomes para melhor leitura do código, e definindo cores personalizadas a serem utilizadas no projeto.

Projeto da Interface:

Perfil do público:

* Jogadores em geral, mas principalmente os mais jovens, que ainda têm pouco contato com games mais atuais de alta tecnologia e complexidade. Também há os jogadores mais velhos, que possuíam a versão física do jogo e gostariam de reviver a experiência do brinquedo.

Telas:

* Menu Principal:

Construída de maneira simplista e fácil de codificar, a tela de menu foi feita pensando na facilidade de uso. O jogador terá apenas de clicar em alguns dos botões coloridos para ir até a tela desejada. Para facilitar ainda mais a compreensão, um pequeno modal exibe um texto dando instruções de como interagir com a tela.

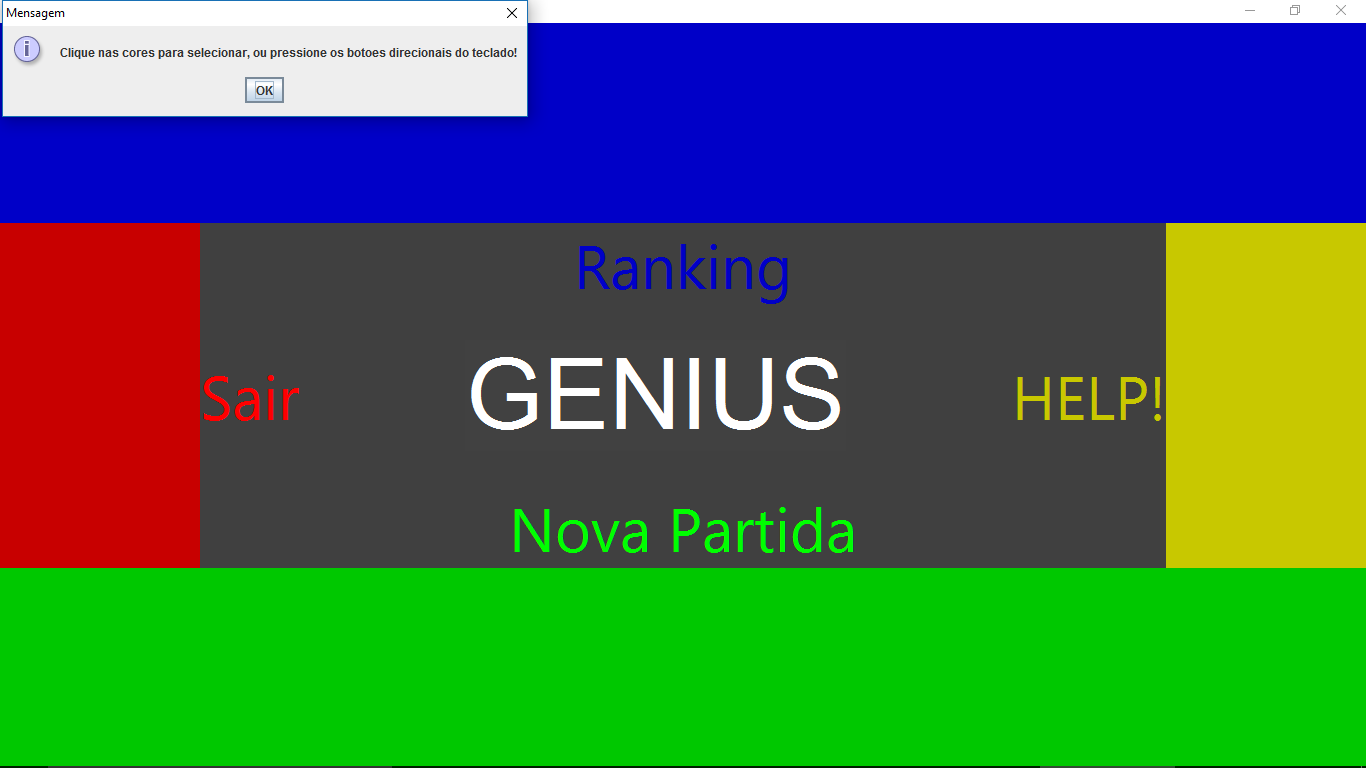


Figura 1 - Menu principal

* Tela de Partida:

Da mesma forma que a tela de Menu, a Tela de Partida, onde realmente ocorre o jogo, foi feita de maneira simplista. Apesar dos botões bem separados, o que dificultaria o uso do mouse, o jogador pode pressionar as teclas direcionais do teclado para interagir. Sempre que necessário, modais com textos explicativos são exibidos para facilitar a experiência.



Figura 2 - Tela onde o jogo acontece

* Tela de Ranking:

Uma tela que exibe dinamicamente as principais pontuações.

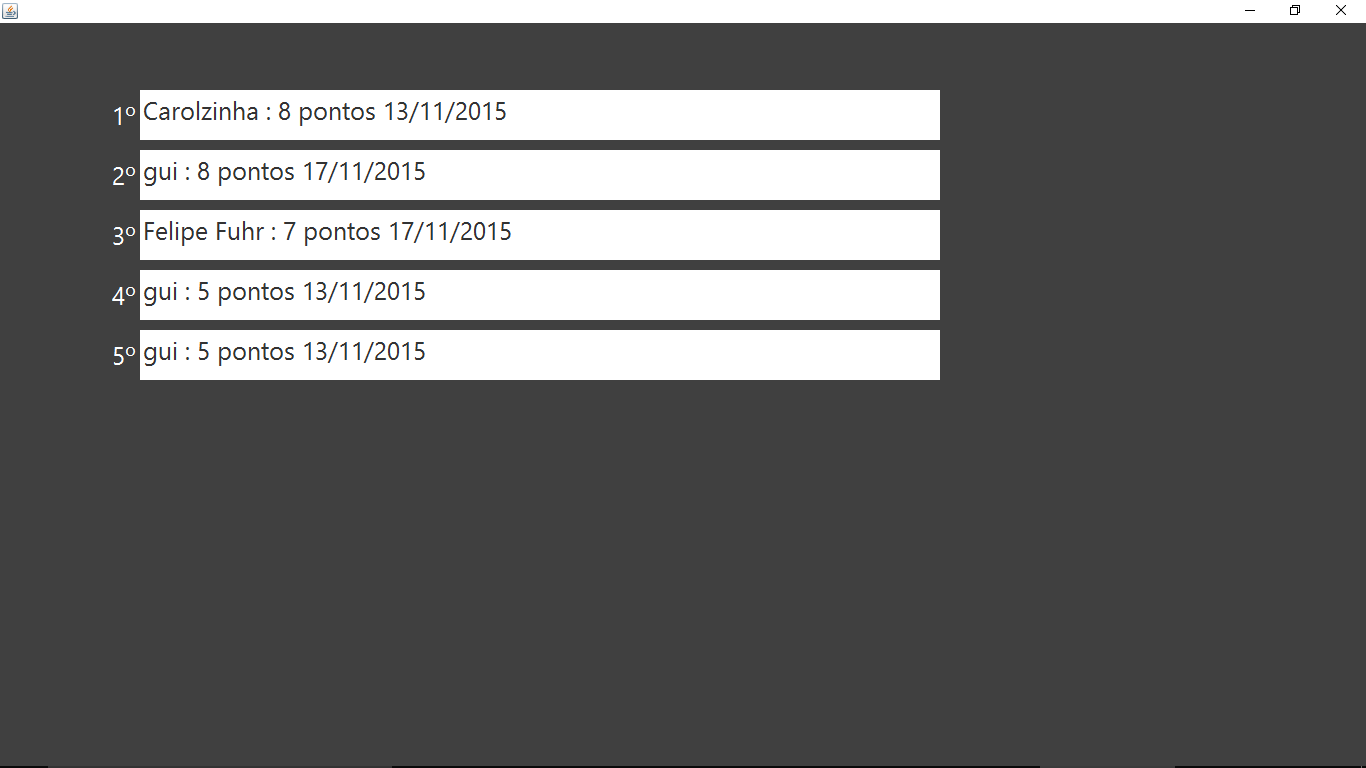


Figura 3 - Exibição das melhores pontuações

* Tela de Help:

Básica como a tela de Ranking, apenas exibe algumas informações úteis para a utilização do software.



Figura 4 - Informações básicas de utilização

* Tela para Inserir Nome:

Mais uma vez, uma tela simples, apenas com título, espaço central para digitação do nome do jogador, e botão chamativo na parte de baixo para iniciar a partida, que também pode ser acionado através da tecla Enter.

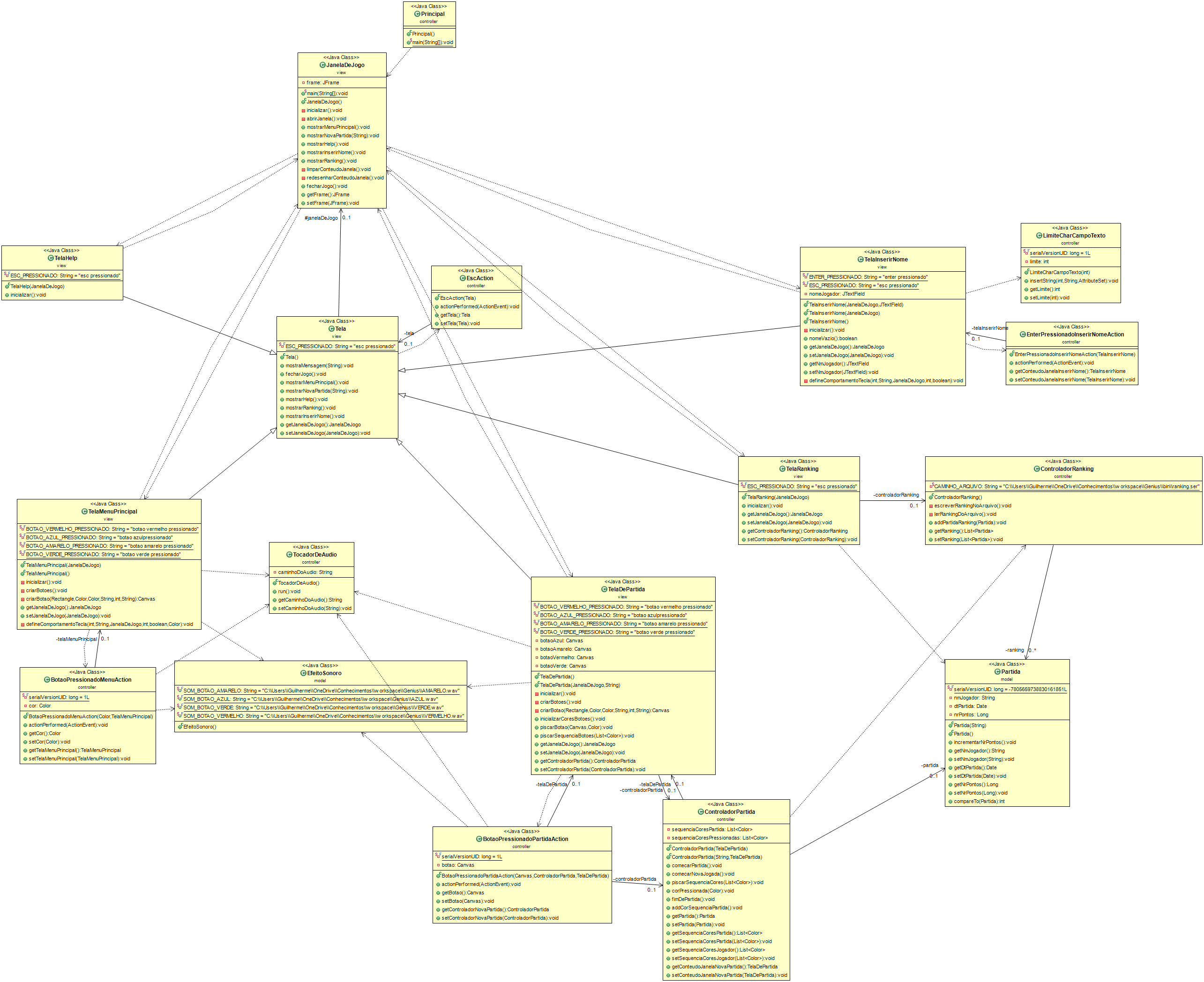


Figura 5 - Inserção do nome do jogador

Outras Classes:

* Actions: BotaoPressionadoMenuAction, BotaoPressionadoPartidaAction, EnterPressionadoInserirNomeAction, EscAction
  + Todas as classes do tipo Action são utilizadas para fornecer o comportamento às teclas do teclado. Cada Classe define o comportamento das teclas e telas referenciadas em seu nome, com exceção da EscAction, que é válida para toda e qualquer tela, e ressaltando que “Botao” faz referência às cores nas telas de Menu e de Partida.
  + Foram adicionadas devido a necessidades técnicas da arquitetura Java Swing e não foram previstas na idealização do projeto.
* LimiteCharCampoTexto:
  + Classe reutilizada de site na internet que define o limite de caracteres a serem inseridos em um campo de texto. Neste projeto em particular, é utilizada na inserção de nome na Tela de Inserir Nome.
  + Foi inserida como ideia durante o processo de construção do código, portanto, também não foi idealizada no início do projeto.
* TocadorDeÁudio:
  + Também baseado em códigos encontrados na internet, é responsável por executar arquivos de áudio em thread separada da original do programa, recebendo apenas o *path* do arquivo.
* Outros Controladores:
  + Certas telas, como InserirNome e MenuPrincipal não necessitaram de controladores, pois seu comportamento pôde ser facilmente definido apenas através de uma action para cada um. A tela de Help, por ser estática, também não possui controlador.

Diagrama de Classes:



Bibliografia:

<http://stackoverflow.com/questions/6045384/playing-mp3-and-wav-in-java> - Código do TocadorDeAudio

<http://stackoverflow.com/questions/3519151/how-to-limit-the-number-of-characters-in-jtextfield> - Código para LimiteCharCampoTexto

<http://docs.oracle.com/javase/7/docs/api/overview-summary.html> - Java Swing Documentation