# Atividade

1. **Faz parte da 2a nota (50%)**
   1. Pode ser feito até em quarteto.
   2. Crie o projeto no modelo do Maven
2. Estratégias para pesquisa
   1. Naive
   2. Rabin Karp
   3. Knuth-Morris-Prat (KMP)
   4. Boyer-Moore
   5. Aho-Corasick
   6. Radix tree
3. Criar uma tela gráfica usando swing que:
   1. Solicite a lista de arquivos txt que serão usados para procura
      1. Exemplo de conteúdo em formato TXT de:
         1. <http://www.folgerdigitaltexts.org/download/>
   2. Solicite uma palavra para o usuário
   3. Realize a procura da palavra usando todas as estratégias
   4. Registre o tempo de execução de cada estratégia
      1. Controle para cada uma delas
         1. Inicio
         2. Fim
         3. Duração
      2. Sugestão: Gravar os resultados em um arquivo CSV (comma separated values)
   5. As estratégias podem ser executadas sequencialmente ou paralelamente
      1. Fica a critério da equipe decidir
   6. Mostre o tempo de execução de cada algoritmo
4. Testar as estratégias

# Entrega

Entregar um artigo com gráficos e comparação de tempo entre as estratégias versus o tipo de entrada

## Prazo

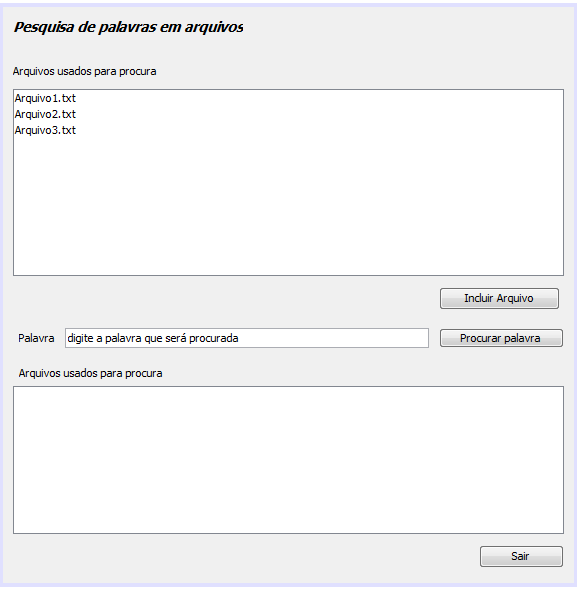
07/05 entrega do material escrito no formato de artigo

## Trabalho

Será feito em sala usando as aulas até a data definida.

*Contínua na próxima página...*

# Exemplo de tela



## Tipos de propriedade para tela

private javax.swing.JButton btnAddFile;

private javax.swing.JButton btnExit;

private javax.swing.JButton btnFindWord;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JLabel labelFiles;

private javax.swing.JLabel labelResult;

private javax.swing.JLabel labelTitle;

private javax.swing.JLabel labelWord;

private javax.swing.JList<String> listFiles;

private javax.swing.JScrollPane scrollFilesPanesl;

private javax.swing.JTextArea textResult;

private javax.swing.JTextField textWord;

## Exemplo de uso

/\*\*

\* Retorna a lista de arquivos

\* @return

\*/

public List<String> getListFiles() {

return listFiles.getSelectedValuesList();

}

/\*\*

\* Define a lista de valores do resultado

\* @param values

\* @return

\*/

public void setTextResultValue(String values) {

textResult.setText(values);

}

/\*\*

\* Pega os valores que já estao na lista

\* @return

\*/

public String getTextResultValue() {

return textResult.getText();

}

/\*\*

\* Retorna a palavra que deve ser procurada

\* @return

\*/

public String getTextWordToFind() {

return textWord.getText();

}

## Base para início

Na pasta da disciplina existe o arquivo procura.zip

## Leitura sugerida sobre MVC

* <https://stackoverflow.com/questions/5217611/the-mvc-pattern-and-swing>
* <https://www.ssaurel.com/blog/learn-to-make-a-mvc-application-with-swing-and-java-8/>

# Leituras sugeridas

Sugere-se as seguintes leituras como apoio

* <https://stackoverflow.com/questions/5217611/the-mvc-pattern-and-swing>
* <https://www.ssaurel.com/blog/learn-to-make-a-mvc-application-with-swing-and-java-8/>
* <http://blog.ivank.net/aho-corasick-algorithm-in-as3.html>
* <http://www.geeksforgeeks.org/longest-prefix-matching-a-trie-based-solution-in-java/>
* <https://stackoverflow.com/questions/9741188/java-indexof-function-more-efficient-than-rabin-karp-search-efficiency-of-text>
* <http://www.geeksforgeeks.org/aho-corasick-algorithm-pattern-searching/>
* <https://www.toptal.com/java/the-trie-a-neglected-data-structure>
* <https://www.toptal.com/algorithms/needle-in-a-haystack-a-nifty-large-scale-text-search-algorithm>
* <https://stackoverflow.com/questions/23336807/when-to-use-rabin-karp-or-kmp-algorithms>