

João Batista Romão Damasceno

Manual de Instalação SwiftLexical

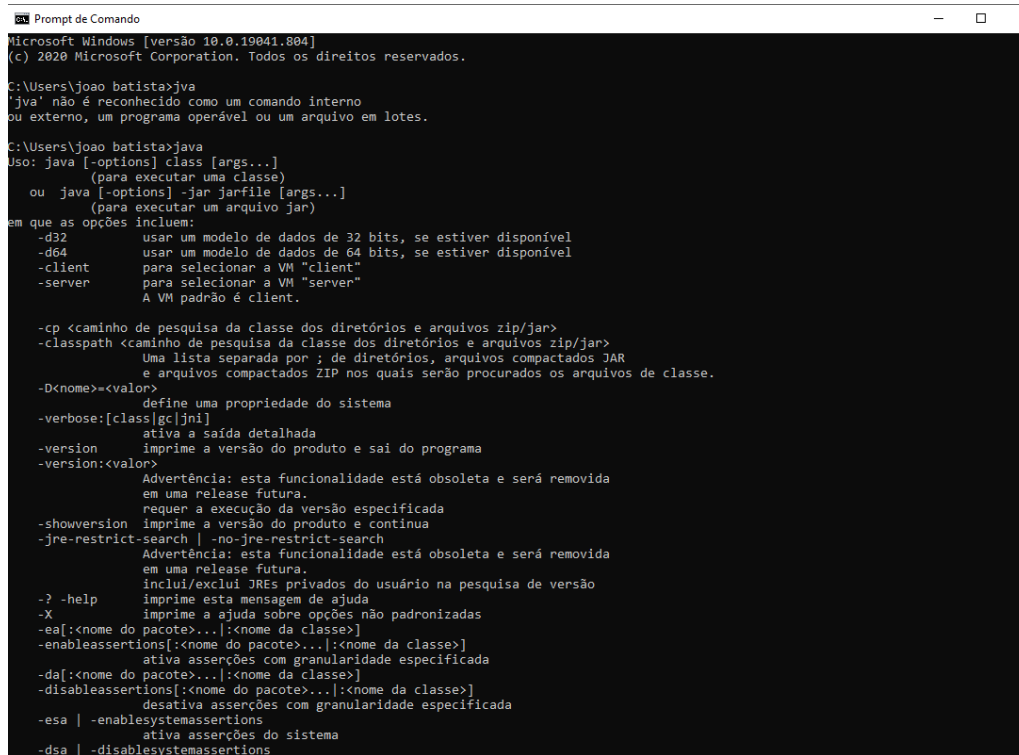


Swift

Manual de instalação do programa
SwiftLexical para o sistema Operacional
Windows desenvolvido para a cadeira de
compiladores do IFCE campus
Maracanaú.

1. Requisitos necessários

Primeiramente, para que se possa instalar a aplicação em sua máquina, é necessário ter o programa Java instalado. Para verificar se o mesmo existe em sua máquina digite na barra de pesquisa do Windows o comando cmd, e ao abrir o terminal digite Java. Caso o terminal retorne algo como na figura 1, isso quer dizer que o Java já está instalado na sua máquina. Caso não você pode acessar esse link: https://www.java.com/pt-BR/download/ie_manual.jsp?locale=pt_BR, lembrando que esse manual é para o sistema operacional Windows, logo, o Java é para Windows.



```
Prompt de Comando
Microsoft Windows [versão 10.0.19041.804]
(c) 2020 Microsoft Corporation. Todos os direitos reservados.

C:\Users\joao batista>java
'java' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

C:\Users\joao batista>java
Uso: java [-options] class [args...]
       (para executar uma classe)
ou java [-options] -jar jarfile [args...]
       (para executar um arquivo jar)
em que as opções incluem:
  -d32      usar um modelo de dados de 32 bits, se estiver disponível
  -d64      usar um modelo de dados de 64 bits, se estiver disponível
  -client   para selecionar a VM "client"
  -server   para selecionar a VM "server"
            A VM padrão é client.

  -cp <caminho de pesquisa da classe dos diretórios e arquivos zip/jar>
  -classpath <caminho de pesquisa da classe dos diretórios e arquivos zip/jar>
            Uma lista separada por ; de diretórios, arquivos compactados JAR
            e arquivos compactados ZIP nos quais serão procurados os arquivos de classe.
  -D<nome>=<valor>
            define uma propriedade do sistema
  -verbose:[class|gc|jni]
            ativa a saída detalhada
  -version   imprime a versão do produto e sai do programa
  -version:<valor>
            Advertência: esta funcionalidade está obsoleta e será removida
            em uma release futura.
            requer a execução da versão especificada
  -showversion
            imprime a versão do produto e continua
  -jre-restrict-search | -no-jre-restrict-search
            Advertência: esta funcionalidade está obsoleta e será removida
            em uma release futura.
            inclui/exclui JREs privados do usuário na pesquisa de versão
  -? -help   imprime esta mensagem de ajuda
  -X         imprime a ajuda sobre opções não padronizadas
  -ea[:<nome do pacote>...[:<nome da classe>]]
  -enableassertions[:<nome do pacote>...[:<nome da classe>]]
            ativa asserções com granularidade especificada
  -da[:<nome do pacote>...[:<nome da classe>]]
  -disableassertions[:<nome do pacote>...[:<nome da classe>]]
            desativa asserções com granularidade especificada
  -esa | -enablesystemassertions
            ativa asserções do sistema
  -dsa | -disablesystemassertions
```

Figura 1 Exemplo prompt de comando com java

Caso queira executar o projeto, é necessário uma IDE de desenvolvimento chamado netbeans, com a versão mínima sendo a 8.2, que pode ser baixada no seguinte link. <https://netbeans.org/downloads/8.2/rc/>. A utilizada nesse projeto foi a última opção com nome tudo.

2. Swift

A linguagem escolhida foi a linguagem swift, que é utilizada para desenvolvimento de aplicativos nativos no ambiente iOS. O motivo de sua escolha é o fato de ser a linguagem a qual trabalho normalmente e por ela é uma linguagem de pouco acesso para os desenvolvedores, em si, pois as aplicações são executadas na IDE Xcode, que só tem para macbooks, o que limita o número de desenvolvedores. Caso queira no ambiente Windows, é necessário um Visual Studio com a versão adequada do xamarin, necessitando bastante de recursos da máquina que nem todos têm o suficiente para aprender de forma mais rápida.

No caso, se tiver um compilador completo para essa linguagem no ambiente windows, a execução e aprendizagem dessa linguagem pode ser bem mais fácil e acessível, além de ser uma linguagem mais simples de entender e interpretar.

3. Como funciona

O analisador foi feito com duas partes principais, a primeira é que gera o analisador em si, a qual o usuário não tem acesso, e a interface UI, que é onde os usuários podem interagir com a aplicação.

Quando foi gerado o programa, ele vem com alguns arquivos e pastas, como a pasta lib aonde é possível encontrar todas as bibliotecas que são necessárias, nesse caso é somente uma a JFlex, três arquivos de exemplo na extensão Swift, e um arquivo do tipo txt com nome arquivo, que é utilizado para guardar o texto da classe que é analisada para que o analisador possa executar.

4. Executando na IDE

Após instalado o java e o netbeans, o projeto pode ser aberto indo na opção arquivos e importar arquivo. Após importado você se depara com a seguinte estrutura de dados apresentada na figura 2

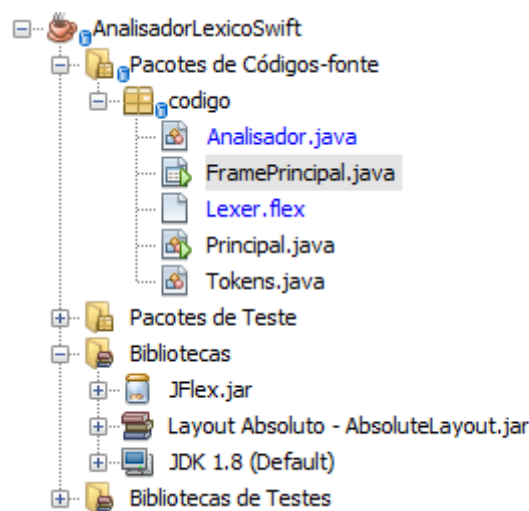


Figura 2 Pastas do projeto

Os únicos arquivos que se deve preocupar são os que estão no pacote código. O primeiro é o analisador gerado automaticamente pelo JFlex, logo não é necessário modificar nada nele. O segundo é a classe responsável pela interface gráfica com o usuário, sendo essa a classe a ser buildada quando for testar a aplicação. Ela contém toda a lógica de negócios da aplicação incluindo leitura e escrita de arquivos e a mesma é responsável por criar uma instancia do analisador e passar as strings para analisar.

Posteriormente tem Lexer.flex, que é o responsável por conter todas as regras para as palavras chaves e textos que existem no código. A classe com nome principal que ela deve ser executada quando há alterações de regras no arquivo flex, sua função é somente ler o arquivo flex, e enviar para a biblioteca JFlex, que vai gerar a nossa classe analisador.

E por fim, temos a classe Tokens, que é uma classe model chamada pelo analisado, para retornar um objeto desse mesmo tipo, que contém o nome do token, o valor, a linha, a coluna e a sua descrição. É esse token gerado que é mostrado na tabela da aplicação.

5. Executaveis

A instalação é bem simples, após baixado a pasta do projeto, você tem duas opções para que possa iniciar a aplicação podendo ser o AnalisadorLexicoSwift, esse tem a extensão .jar, pois ele é gerado nativamente na IDE netbeans, já o outro é o SwiftLexical, que é um executável para a plataforma Windows com sistema operacional 64 bits.

A diferença entre os dois é que o AnalisadorLexicoSwift precisa ter a pasta lib, com a biblioteca JFlex, para que possa funcionar de forma correta e o SwiftLexical não, o executável já tem essa biblioteca dentro, então fica mais fácil enviar. A figura 3 mostra a baixa baixada e as duas opções de uso da aplicação.

Nome	Data de modificação	Tipo	Tamanho
lib	22/02/2021 20:38	Pasta de arquivos	
AnalisadorLexicoSwift	22/02/2021 20:38	Executable Jar File	67 KB
arquivo	22/02/2021 21:04	Documento de Te...	1 KB
README	22/02/2021 20:38	Documento de Te...	2 KB
teste salvar	22/02/2021 20:39	Documento de Te...	1 KB

Figura 3 Pasta com aplicação

Se tudo ocorrer de acordo o resultado ao executar a aplicação pode ser visto na figura 4. Nela encontramos as funções principais do app que é analisar um código importado ou digitado na área de texto, e apagar a análise gerada.

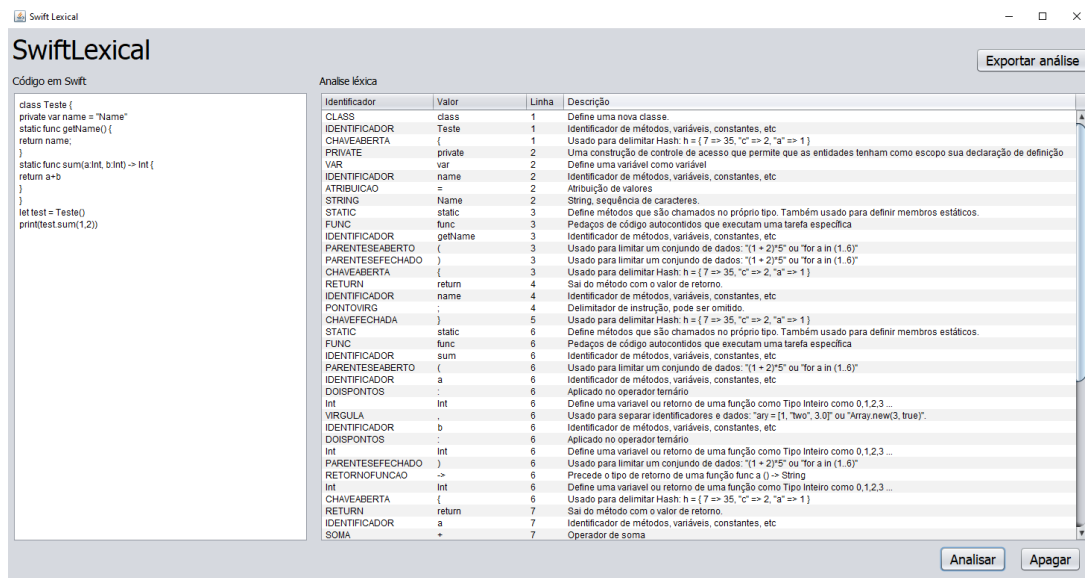


Figura 4 Aplicativo em execução

6. Cuidados necessários

Caso utilize o AnalisadorLexicoSwift, ele não pode sair de dentro da pasta a qual recebeu, pois ela contém os arquivos necessários para o seu bom funcionamento. Alterações do arquivo de biblioteca também não é recomendado. Todo arquivo que for usado para a leitura de dados deve estar na mesma pasta que o arquivo em execução para que o mesmo

possa encontra-lo ou em uma subpasta, com o caminho correto. E a extensão do arquivo tem que ter a extensão swift.