|  |  |
| --- | --- |
|  | **UNIVERSIDADE DO OESTE PAULISTA**  **CIÊNCIA DA COMPUTAÇÃO** |

**GABRYEL HENRIQUE BORGES**

**LEON BRUCHMANN RONCHI**

**DISCIPLINA**

Presidente Prudente - SP

2024

1. **Introdução**

No projeto, foi desenvolvido um simulador capaz de lidar com gramáticas regulares e expressões regulares. A escolha do JavaFX como framework para a interface gráfica permitiu a criação de uma experiência interativa para o usuário.

O projeto foi dividido em diferentes controladores e entidades para uma clara separação de responsabilidades:

* **Controladores**: Responsáveis pela interação com o usuário e pelo fluxo da aplicação, como GramRegViewController e ExpRegViewController.
* **Entidades**: Modelos que representam os conceitos principais, como gramáticas, por exemplo, a classe Gramatica, sendo possível exibir a execução passo a passo encontrada. Na expressão regular não foi necessário o uso de entidade, pois o uso da biblioteca Matcher permitiu que a entrada fosse verificada automaticamente.

**2. Funcionalidades Implementadas**

**2.1 Simulador de Gramáticas Regulares**

A interface permite que o usuário insira variáveis, símbolos terminais, regras de produção e um símbolo inicial, conforme a gramática regular especificada (G = V T P S). A lógica de verificação percorre uma cadeia de entrada e garante que a gramática é válida ao seguir as regras de produção.

* **Verificação Passo a Passo**: Esta funcionalidade permite ao usuário ver cada etapa da derivação da cadeia feita a partir das regras de produção.
* **Função de Verificação**: A função verificaGramatica faz a validação se a cadeia pode ser gerada pela gramática, verificando a consistência entre os símbolos terminais e as regras de produção.

**2.2 Simulador de Expressões Regulares**

O simulador aceita entradas que consistem em caracteres alfanuméricos, operadores de união (|), concatenação, fechamento (\*), e o símbolo ε para vazio.

* **Validação de ER**: A função ehExpRegValida verifica se a expressão regular segue o formato esperado.
* **Testes com Entradas**: A função testaExpReg permite ao usuário testar se uma dada expressão regular aceita ou rejeita uma cadeia de entrada, fornecendo feedback claro por meio de uma mensagem.

**3. Qualidade da Solução**

A estruturação do código segue boas práticas de programação:

* **Coesão e Separação de Responsabilidades**: Cada classe possui uma única responsabilidade, garantindo que o código seja modular e fácil de entender. Os controladores tratam diretamente da interface e da lógica de interação, enquanto as entidades como Gramatica contêm a lógica interna do processamento.
* **Tratamento de Erros e Mensagens ao Usuário**: A utilização da classe utilitária Util facilita o gerenciamento de feedback ao usuário, garantindo uma experiência clara e amigável. O uso de mensagens de alerta ao detectar erros, como gramáticas ou expressões regulares inválidas, melhora a usabilidade.
* **Uso de Listeners**: A desativação automática dos botões de ação, quando os campos de texto estão vazios, melhora a experiência de usuário e evita erros de execução desnecessários.

**4. Conclusão**

A solução implementada atende aos requisitos do simulador de linguagens formais, proporcionando uma interface intuitiva para a definição e simulação de gramáticas regulares e expressões regulares. O uso de JavaFX facilitou a criação de uma interface gráfica responsiva e funcional.