GERAÇÃO DE PROGRAMAS PARA TESTES DE APIS DE LINGUAGENS DE PROGRAMAÇÃO

CASSIANE VITÓRIA GAIARDO[1], JOÃO PEDRO LAMAISON MIRANDA[2], SAMUEL DA SILVA FEITOSA[3]

1 Introdução

O Java é uma linguagem de programação de alto nível, orientada a objetos e fortemente tipada. Foi criada em 1995 e é considerada antiga pelos usuários, no entanto está longe de cair em desuso, conforme aponta uma pesquisa feita pelo GitHub que a elegeu como terceira linguagem mais utilizada tanto em 2021 quanto em 2022 perdendo apenas para o JavaScript e o Python (ALECRIM, 2023). Ou seja, continua sendo popular e recebe atualizações constantes.

Em 2014 foi lançada a versão 8 do Java e nela foram introduzidos conceitos de programação funcional, que prezam pela diminuição da escrita, melhor legibilidade e códigos menos repetitivos. As expressões lambda são um meio de tornar a linguagem mais funcional, as quais podem ser vistas como métodos anônimos, ou seja, aqueles que não possuem nome, além disso possuem parâmetros e definição de tipos opcionais. As informações dos tipos são inferidas automaticamente pelo compilador de acordo com o contexto. São excelentes como funções simples que serão usadas apenas uma vez.

O gerador desenvolvido teve início no IFSC e produz códigos bem tipados e aleatórios, dentre as quais invocação de métodos, criação de objetos, conversões de tipos, geração de acesso a propriedades, entre outras. (KRAUS, SCHAFASCHEK E FEITOSA, 2021).

Tendo em vista esta atualização juntamente com o projeto do IFSC, o trabalho atual é uma aplicação do gerador de códigos aleatórios na geração de expressões lambda. Para permitir isso foram adaptadas e implementadas novas funcionalidades com objetivo de permitir a programação funcional sem interferir em nenhuma das funções anteriores.

[1]Técnica em Informática IFRS e graduanda de Ciência da Computação UFFS, cassiane.cvg@gmail.com
[2] Graduando, Universidade Federal da Fronteira Sul Chapecó
[3] Doutor, Universidade Federal da Fronteira Sul Chapecó

2 Objetivos

O objetivo principal é aprimorar e adaptar o projeto para que consiga produzir expressões lambda corretamente a partir do que está sendo fornecido pelo gerador de código aleatório e assim garantir integração da programação funcional.

2.1 Objetivos específicos

- Conhecimento bibliográfico sobre o assunto;
- Compreensão do projeto como todo;
- Estimular pesquisas sobre o assunto/ área;
- Ampliação e inclusão de novas funcionalidades no gerador.

3 Metodologia

Primeiramente foi realizada uma ambientação do projeto existente, leituras, resumos, pesquisas bibliográficas e uma série de testes com a finalidade de situar na compreensão do que é o projeto e o que ele é capaz de produzir.

Na sequência iniciou-se a implementação, nesta etapa foram aplicados os conceitos obtidos anteriormente como o de interfaces funcionais, que são interfaces que possuem apenas um método abstrato. Além disso, foi necessário separar os métodos em concretos e candidatos a lambda, ou seja, isolar os abstratos provenientes de uma interface funcional, dos demais métodos não abstratos. Para só então produzir a expressão lambda seguindo a sintaxe padrão e invocar a expressão através do método candidato anteriormente citado.

Tendo isso concluído, inicia-se a fase de testes. O objetivo dos testes é obter dados concretos e garantir que o resultado é confiável. Para isso serão realizados testes baseados em propriedades, verificando cada uma das funções separadamente e em aplicação usando diferentes compiladores. Serão resolvidos quaisquer eventuais erros e inconsistências para garantir assim a confiabilidade do gerador.

4 Resultados Parciais e Discussão

A geração das expressões lambda contam com uma sintaxe padrão (argumento) -> (corpo). O argumento seria o valor de entrada da função, pode ter mais de um ou mesmo nenhum parâmetro, mas nesse caso o argumento produzido é um valor único e apesar de ser possível inferir o tipo, ele é declarado explicitamente. Já o corpo seria o conteúdo que ficaria

dentro das chaves em uma função padrão, gera expressões válidas que variam, entre outros fatores, na criação de objetos e acessando métodos/ campos, ou números e expressões numéricas. Exemplo:

```
(double a) -> new br.edu.ifsc.javargexamples.A(-527, -139051, false).getA1()
(double d) -> 126
```

Imagem 1: Exemplo expressão lambda

Neste exemplo podemos visualizar duas expressões lambda formadas pelo gerador, nota-se que o corpo da primeira é composto por um objeto da classe A passando três parâmetros e acessando o método *getA1*. Enquanto o segundo é apenas um valor numérico inteiro.

Depois de ter essa parte concluída, resta invocar a expressão. E para isso precisamos identificar os métodos abstratos válidos.

```
public List<Method> getLambdaCandidateMethods(String type)
        throws ClassNotFoundException {
    List<Method> candidatesMethod = new ArrayList<>();

    for (String c : mImports) {
        if (Class.forName(c).isInterface()) {
            List<Method> mthd = getClassMethods(c);

            List<Method> collect = mthd
                    .stream()
                    .filter(m -> m.getReturnType().toString().equals(type)
                    && Modifier.isAbstract(m.getModifiers()))
                    .collect(Collectors.toList());

            if (collect.size() == 1) {
                candidatesMethod.addAll(collect);
            }
        }
    }
    return candidatesMethod;
}
```

Imagem 2: Função métodos candidatos a lambda.

Pode-se notar que a função acima possui uma condição *isInterface*, um filtro *isAbstract* e outra condição verificando se a quantidade de métodos é 1 (um), assim garantindo que é um candidato possível a lambda. De maneira análoga, porém inversa, ocorreu com os métodos concretos, que possuem um filtro que garante que não há métodos abstratos entre eles.

Ao invocar a expressão junta-se as duas partes, expressão lambda e método candidato a lambda, finalizando a geração e iniciando a fase de testes e ajustes finais. Exemplo:

```
(double b) -> new br.edu.ifsc.javargexamples.A(-2, 2147483647, false).getA2().applyAsInt(b)
```

Imagem 3: Código gerado para invocar a expressão lambda.

5 Conclusão

Neste trabalho pode-se observar a implementação da programação funcional no gerador de códigos aleatórios por meio das expressões lambda. Para atingir esse objetivo foram conduzidas extensas pesquisas bibliográficas e diversos conceitos derivados foram abordados. Espera-se que a partir deste estudo outros possam ter início, assim como a continuação das aplicações e melhorias no gerador de código, afinal ainda há muito que se pode abordar sobre o tema.

Como trabalho futuro pode-se ampliar ainda mais as ferramentas do gerador, alguns exemplos incluem o construtor *switch* que teve diversas atualizações interessantes desde seu lançamento, ou as classes *records* que poupariam o gerador de escrever muito código, ou ainda incluir controle de herança com as classes *seladas*, são várias as opções que valem ser incluídas fazendo com que os códigos e os testes gerados sejam cada vez mais completos.

Referências Bibliográficas

KRAUS, Luiz Felipe et al. Synthesis of Random Real-World Java Programs from Preexisting Libraries. In: PROCEEDINGS of the 25th Brazilian Symposium on Programming Languages. Joinville, Brazil: Association for Computing Machinery, 2021. (SBLP ’21), p. 108–115.

FEITOSA, Samuel; RIBEIRO, Rodrigo; DU BOIS, André. A Type-Directed Algorithm to Generate Well-Typed Featherweight Java Programs: 21st Brazilian Symposium, SBMF 2018, Salvador, Brazil, November 26–30, 2018, Proceedings. In: [s.l.: s.n.], jan. 2018. P. 39–55

KRAUS, Luiz; SCHAFASCHEK, Bruno; FEITOSA, Samuel. Desenvolvimento de um Gerador de Programas Aleatórios em Java. In: p. 485–487.

ALECRIM, Emerson. Github: JavaScript, Python e Java foram as linguagens mais usadas em

2022. Tecnoblog, 01 fev. 2023. Disponível em: https://tecnoblog.net/noticias/2023/02/01/github-javascript-python-e-java-foram-as-linguagens-mais-usadas-em-2022/ . Acesso em: 22 jul. 2023.

COELHO, Bruno Schafaschek; KRAUZ, Luiz Felipe; FEITOSA, Samuel da Silva. Desenvolvimento de uma Ferramenta para Teste Diferencial de Compiladores Usando Códigos Gerados Aleatoriamente. *In*: WORKSHOP-ESCOLA DE INFORMÁTICA TEÓRICA (WEIT), 6. , 2021, Online. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2021 . p. 68-72.

Palavras-chave: Geração de Programas Aleatórios; Expressões Lambda; Teste Baseado em Propriedades;

Nº de Registro no sistema Prisma: PES 2022 - 0123

Financiamento

Este projeto foi desenvolvido com apoio financeiro para o pagamento de bolsas de pesquisa pela Universidade Federal da Fronteira Sul - UFFS, através do edital nº 89/GR/UFFS/2022.