Prática 07 - Teste Baseado em Propriedades

Breno Farias da Silva

2025-06-09

## Introdução

Este relatório descreve a aplicação da técnica de **teste baseado em propriedades (Property-Based Testing)** para verificar a implementação de uma função que valida identificadores na linguagem *Silly Pascal*. A atividade faz parte da disciplina **PPGCC12 - Teste de Software**.

A função validateIdentifier(String s) possui o objetivo de determinar se um identificador é válido segundo as seguintes regras:

* Deve começar com uma letra (maiúscula ou minúscula).
* Pode conter apenas letras e dígitos após o primeiro caractere.
* O comprimento deve ser de **1 a 6 caracteres**, inclusive.

## Análise da Solução Implementada

A análise do código fornecido revela alguns problemas. O trecho abaixo mostra a implementação:

public boolean validateIdentifier(String s) {  
 char achar;  
 boolean valid\_id = false;  
 achar = s.charAt(0);  
 valid\_id = ((achar >= 'A') && (achar <= 'Z')) || ((achar >= 'a') && (achar <= 'z'));  
  
 if (s.length() > 1) {  
 achar = s.charAt(1);  
 int i = 1;  
 while (i < s.length() - 1) {  
 achar = s.charAt(i);  
 if (((achar >= 'A') && (achar <= 'Z')) || ((achar >= 'a') && (achar <= 'z')) || ((achar >= '0') && (achar <= '9'))) {  
 valid\_id = false;  
 }  
 i++;  
 }  
 }  
 if (valid\_id && (s.length() >= 1) && (s.length() < 6))  
 return true;  
 else  
 return false;  
}

### **Observações sobre o código**

* Há um erro lógico no laço while: a condição invalida o identificador sempre que um caractere válido (letra ou dígito) é encontrado. Isso é oposto ao esperado.
* O limite de tamanho está incorreto, pois o código verifica s.length() < 6, quando deveria ser <= 6.
* Não há tratamento para strings vazias, o que pode gerar exceção (StringIndexOutOfBoundsException).

## Definição das Propriedades

Baseado na especificação, definimos as seguintes propriedades para a função:

### **Propriedades de entrada válida**

* Para qualquer string que:
  + Tenha de 1 a 6 caracteres;
  + O primeiro caractere seja uma letra ([a-zA-Z]);
  + Os demais caracteres (se existirem) sejam letras ou dígitos ([a-zA-Z0-9]);

A função **deve retornar true**.

### **Propriedades de entrada inválida**

* A função deve retornar **false** se:
  + O primeiro caractere não for uma letra;
  + O comprimento for menor que 1 ou maior que 6;
  + Contiver qualquer caractere que não seja letra ou dígito.

## Implementação dos Testes Baseados em Propriedades

A implementação dos testes foi realizada utilizando o framework **jqwik** para Java. Foram criados três conjuntos de propriedades:

### **Testa identificadores válidos**

@Property  
void validIdentifiers(@ForAll @AlphaChars @StringLength(min = 0, max = 5) String suffix) {  
 String id = "A" + suffix;  
 assumeTrue(id.length() <= 6);  
 assertThat(identifier.validateIdentifier(id)).isTrue();  
}

### **Testa identificadores inválidos por tamanho**

@Property  
void invalidLengthIdentifiers(@ForAll @StringLength(min = 7, max = 20) String id) {  
 assertThat(identifier.validateIdentifier(id)).isFalse();  
}

### **Testa identificadores inválidos por caracteres**

@Property  
void invalidCharacterIdentifiers(@ForAll @StringLength(min = 0, max = 5) String suffix) {  
 String id = "1" + suffix;  
 assumeTrue(id.length() <= 6);  
 assertThat(identifier.validateIdentifier(id)).isFalse();  
}

## Resultados dos Testes

| Propriedade | Status | Casos Gerados |
| --- | --- | --- |
| Identificadores válidos | Falha | 1 (primeiro caso) |
| Identificadores inválidos por tamanho | Passou | 1000 |
| Identificadores inválidos por caract. | Passou | 1000 |

### **Detalhes da falha**

* A propriedade **“Identificadores válidos”** falhou logo na primeira execução.
* Casos que falharam: "AA" e "YxU".
* Erro: A função retornou false para identificadores válidos.

## Conclusão

O teste baseado em propriedades se mostrou eficiente na identificação de problemas na implementação. A falha detectada reforça que a função não atende corretamente à especificação quando se trata de validar identificadores válidos.

A técnica não só verifica casos simples, mas também explora automaticamente cenários de borda e aleatórios, proporcionando uma cobertura ampla e eficaz.

### 🔧 **Recomendação**

A implementação fornecida apresenta erros lógicos. Recomenda-se a sua correção conforme a especificação. Segue uma versão correta da função:

public boolean validateIdentifier(String s) {  
 if (s == null || s.length() < 1 || s.length() > 6)  
 return false;  
  
 if (!Character.isLetter(s.charAt(0)))  
 return false;  
  
 for (int i = 1; i < s.length(); i++) {  
 char c = s.charAt(i);  
 if (!Character.isLetterOrDigit(c))  
 return false;  
 }  
  
 return true;  
}

## Referências

* Livro: *Effective Software Testing* — Maurício Aniche, 2022.
* Documentação jqwik: https://jqwik.net/docs/current/user-guide.html