Relatório Trabalho Prático de Teste Baseado em Especificação

Validação de identificadores no Silly Pascal

Breno Farias da Silva

Sumário

Introdução

Este documento descreve o processo utilizado para a elaboração de testes baseados em especificação, aplicados ao método `validateIdentifier(String s)`, responsável por validar identificadores na linguagem *Silly Pascal*. O objetivo é identificar partições de equivalência, valores-limite e derivar casos de teste de forma sistemática com base nas regras de negócio fornecidas.

Passos para execução do teste efetivo e sistemático

Passo 1 - Explorar o funcionamento do programa

O método `validateIdentifier(String s)` deve verificar se uma string representa um identificador válido em Silly Pascal, obedecendo às seguintes regras:

1. Conter entre 1 e 6 caracteres;
2. Iniciar obrigatoriamente com uma letra (maiúscula ou minúscula);
3. Conter apenas letras e dígitos (não são permitidos espaços, símbolos ou outros caracteres especiais);

Com base nessas regras, é possível identificar entradas válidas e inválidas a partir da análise do comportamento esperado, sem depender da lógica interna do método.

Passo 2 - Identificar as partições

Regras de entrada

- Comprimento da string:
  - comprimento < 1 (inválido)
  - comprimento entre 1 e 6 (válido)
  - comprimento > 6 (inválido)
- Primeiro caractere:
  - letra (válido)
  - não letra (inválido)
- Demais caracteres:
  - todos são letras ou dígitos (válido)
  - pelo menos um caractere inválido (inválido)

Combinando partições

Com base nas regras, temos as seguintes partições significativas:

1. Vazio (`""`) – inválido por comprimento
2. Uma letra (`"a"`) – válido
3. Uma letra seguida de letras/dígitos válidos (`"abc123"`) – válido
4. Começa com número (`"1abc"`) – inválido
5. Caracteres especiais (`"a@c"`) – inválido
6. Excede 6 caracteres (`"abcdefg"`) – inválido
7. Letra seguida de espaços ou outros símbolos (`"a b"`) – inválido

Passo 3 - Identificar os valores limite

- Limite inferior: comprimento 1 (válido), comprimento 0 (inválido)
- Limite superior: comprimento 6 (válido), comprimento 7 (inválido)
- Limite para caracteres válidos: letra (válido), dígito (válido), símbolo (inválido)

Passo 4 - Derivar os casos de teste

Adicionalmente, foi considerado o caso de strings que possuem apenas espaços em branco. Apesar de não serem literalmente vazias (como `""`), elas ainda assim não satisfazem a regra de conter um identificador válido, e devem ser tratadas como inválidas.

Caso	Entrada	Esperado	Justificativa
1	“”	false	Comprimento 0 (inválido)
2	“a”	true	Letra única, válido
3	“abc123”	true	6 caracteres válidos
4	“abcdefg”	false	Mais de 6 caracteres
5	“1abc”	false	Primeiro caractere não é letra
6	“a b”	false	Espaço é inválido

Caso	Entrada	Esperado	Justificativa
7	“ab@1”	false	Símbolo ‘@’ inválido
8	“abc12”	true	Dentro do limite e todos válidos
9	“ab1c!”	false	Exclamação é inválida
10	“AbC9”	true	Letras e dígitos, caso misto
11	” ”	false	String com espaços, mas sem caracteres válidos

Análise dos resultados dos testes

Os testes foram executados utilizando o framework JUnit, por meio do comando `mvn test`. A execução resultou em 4 falhas e 1 erro, conforme indicado no relatório do Maven. Esses resultados evidenciam que a implementação atual do método `validateIdentifier` **não está em conformidade com as regras especificadas**. Abaixo, detalhamos os casos problemáticos:

- **`testEmptyString`**
  - **Entrada:** `""` (string vazia)
  - **Esperado:** `false`
  - **Obtido:** Erro de execução (`StringIndexOutOfBoundsException`)
  - **Comentário:** O método não trata o caso de string vazia, acessando diretamente o índice 0 sem verificação prévia.
- **`testContainsSpace`**
  - **Entrada:** `"abc 123"`
  - **Esperado:** `false`
  - **Obtido:** `true`
  - **Comentário:** A função aceita espaços como válidos, contrariando a especificação que proíbe qualquer caractere que não seja letra ou dígito.
- **`testValidIdentifierMaxLength`**
  - **Entrada:** `"abcde"`
  - **Esperado:** `true`
  - **Obtido:** `false`
  - **Comentário:** Um identificador com 5 letras, que deveria ser aceito, foi incorretamente rejeitado.
- **`testMixedCaseWithDigit`**
  - **Entrada:** `"AbC9"`
  - **Esperado:** `true`
  - **Obtido:** `false`
  - **Comentário:** A função não reconhece corretamente combinações válidas de letras maiúsculas/minúsculas com dígitos.
- **`testValidIdentifierShort`**
  - **Entrada:** `"abc12"`

- **Esperado:** `true`
- **Obtido:** `false`
- **Comentário:** Um identificador curto e válido foi erroneamente marcado como inválido.

Considerações finais

A técnica de teste baseado em especificação permite a derivação de casos de teste coerentes e eficazes a partir das regras funcionais declaradas, sem a necessidade de conhecer a implementação do método. Essa abordagem auxilia na validação de comportamentos esperados e na verificação da conformidade com os requisitos da linguagem, contribuindo significativamente para a garantia da qualidade do software.

Esse resultado indica que a função `validateIdentifier` possui problemas tanto de lógica quanto de robustez, já que retorna valores inesperados em alguns casos e lança exceção ao receber uma string vazia.