

Hangman Java

Autores:

Eric de Jesus Machado - 201703749

Cauã dos Santos Rebelo - 201703743

Jair Souza Meira Rodrigues - 201910884

Teste e Prática(s) de Teste Organizacional

Política(s) de Teste

Objetivos de testagem: Tem como objetivo a garantia que o jogo será testado pela E.C Soft LTDA e que possa então funcionar conforme planejado no projeto inicial.

Processo de teste: Todos os testes conduzidos serão baseados nas Especificações do Projeto de Testes feitos pelo professor Cássio Leonardo Rodriguez. Os testes incluem: Teste de Unidade feito no JUnit, Teste de Mutação no PIT test e Teste de Aceitação feito no Cucumber.

Organização dos testes: os testes serão realizados de maneira sequencial, sendo os Testes de Unidades, Testes de Aceitação e por último os Testes de Mutação.

Código de ética: Todos os testadores devem aderir ao código de ética da companhia.

Padrões: A organização em questão, segue os padrões estabelecidos pela ISO/IEC 20246 (quando se trata de testes estáticos), ISO/IEC/IEEE 29119-2, ISO/IEC/IEEE 29119-3 e ISO/IEC/IEEE 29119-4.

Aprimoramento do processo de teste e determinação de valor: obrigatoriamente, deve-se prover relatórios com dados coletados em diversos pontos do ciclo de desenvolvimento e testes a fim de aprimorar o processo, boas práticas e normas seguidas pelos testadores.

Assim, ao ter embasamento necessário, são estudados e estabelecidos objetivos de melhoria nos pontos identificados.

Armazenamento e reuso de artefatos de teste: obrigatoriamente, deve-se manter registro de todos os processos, boas práticas, normas e artefatos de teste à disposição, com a companhia provendo repositórios em sua rede privada com os devidos cuidados em relação à segurança da informação.

Prática(s) de Teste Organizacional

Critérios de entrada: Em relação ao Plano de Teste do Projeto, têm-se alguns dos itens necessários para os critérios de entrada nos testes de sistema, unidade e integração, estático, aceitação, verificação de produção, desastres, penetração e de desempenho: os requisitos de cada teste, definições do ambiente de teste, verificação de abrangimento nos estágios anteriores de teste, os casos de teste devem ser projetados, rastreados os requisitos é assinado pelas partes interessadas.

Critérios de saída: em testes estáticos, deve-se produzir e atentar aos requisitos do projeto e inspeções de código.

Critérios de conclusão do teste: para se ter como concluídos, o projeto deve ter 100% cobertura em relação a testes, além disso, deve sem exceção, solucionar e realizar o novamente os testes de erros de média e alta gravidade previamente encontrados.

Documentação e relatórios de teste: deve-se atentar ao produzir a documentação e os relatórios de testes pautados na ISO/IEC/IEEE 29119-3 em todos os casos de aplicação de testes dinâmicos e o ISO/IEC 20246 em todos os casos de aplicação de testes estáticos.

Grau de independência: a fim de garantir independência na prática de teste, os testadores devem seguir a hierarquia estabelecida pela companhia e reportar ao seu gerente de projeto e ao chefe de teste.

Técnicas de projeto de teste: essas técnicas devem ser implementadas para cobrir os requisitos e decrescer riscos, em adendo, adaptações impostas na norma definida pela companhia deve ser aprovada e assinada pelo Chefe de Desenvolvimento e Chefe de Testes.

Adicionalmente, as especificações das técnicas e estrutura de execução definidas pela ISO/IEC/IEEE 29119-4. Ambiente de teste e métricas coletadas: o ambiente de teste deve reproduzir com o máximo de fidelidade ao cenário real do projeto, buscando fidelidade nos dados a se obter com os testes.

Na impossibilidade de máxima fidelidade, faz-se necessário o uso de maneiras de simulação aproximada da realidade. Em relação às métricas, deve-se relatar durante os testes as métricas, tais como: número de erros de produção normalizado pelo tamanho do projeto, porcentagem de defeitos que surgiram por não serem abordados em uma etapa de teste anterior e o número esperado e real de testes projetados e executados por dia.

Reteste e teste de regressão: no caso de repetição do teste, após corrigir-se os defeitos encontrados em casos de teste anteriores, os casos de teste devem ser executados novamente para verificar se foram de fato abrangidos ou ainda estão defeituosos.

Em relação ao teste de regressão, os casos de teste devem ser obrigatoriamente executados após os níveis de Teste de Sistema e Teste de aceitação, com o nível de Teste de Sistema ficando a cargo da decisão de necessidade do Chefe de Teste e do Gerente de Projeto.

Descrição do Software

O jogo Hangman é um simples jogo que tem como objetivo que um jogador insira uma palavra, que fica escondido aos outros jogadores, então cada jogador tenta adivinhar uma letra apenas sabendo a quantidade de letras que a palavra tem, se eles tentam adivinhar uma letra que não está na palavra eles perdem uma tentativa, os jogadores vencem se adivinharem a palavra inteira antes de perderem todas as

tentativas. Logicamente os jogadores perdem se esgotarem todas as tentativas antes de acertarem a palavra.

Plano de teste

Objetivo: Este documento tem o objetivo de descrever o plano para testes do projeto de arquitetura do Hangman Game. Este documento de Plano de Teste suporta os seguintes objetivos: Executar diversos testes no projeto Hangman Game, para buscar possíveis falhas, vulnerabilidades e bugs.

Escopo: Este Plano de Teste descreve os testes unitário, cobertura, mutação e aceitação que serão conduzidos no projeto de testes.

Estratégia do Teste: Clonando o projeto do GitHub Hangman Game, fizemos modificação para conseguirmos rodar os testes requeridos. Para o quesito de teste, consideramos somente a Classe Game pois as classes Hagman e Prompter não tem sentido se utilizadas sozinhas pois são dependentes da Classe principal Game. Considerando todos os fatos citados, utilizamos JUnit, Cucumber e Pit Mutation Test, para teste de unidade, teste de aceitação e teste de mutação respectivamente.

Ferramentas:

Tipo de teste	Ferramenta	Versão
Teste unitário	JUnit	4.12
Teste de aceitação	Cucumber	7.5.0
Teste de mutação	PIT Mutation test	1.6.8

Cronograma de testes:

Tarefa	Esforço	Início	Encerramento
--------	---------	--------	--------------

Teste unitário	4	19/08/2022	26/08/2022
Teste de aceitação	3	02/09/2022	05/09/2022
Teste de mutação	5	26/08/2022	02/09/2022

Casos de teste de Unidade:

Em todos os casos de teste a palavra esperada é “hello”.

CT01: Testar com entrada ‘h’ ‘e’ ‘l’ ‘o’ e vencendo.

CT02: Testar com entrada ‘o’ ‘b’ ‘d’ ‘g’ ‘y’ ‘r’ ‘t’ buscando falhar no jogo.

CT03: Testar com entrada ‘w’ e ver se o jogo terminou’.

CT04: Testar com entrada ‘h’ e terminando o jogo sem adivinhar.

CT05: Testar com entrada [null], buscando um erro de exceção.

CT06: Testar com entrada “ vazia, buscando um erro de exceção.

CT07: Testar com entrada ‘2’ buscando exceção por não ser uma letra.

CT08: Testar com entrada “h” e verificar se faltam 4 letras restantes.

CT09: Testar com entrada ‘h’ ‘h’ para verificar se ocorreu erro por inserir a mesma entrada.

CT10: Testar qual a resposta do jogo da forca.

Casos de teste	Valores de entrada	Resultado esperado
CT 01	h e l o	6 tentativas restantes, jogo resolvido

CT 02	o b d g y t r	0 tentativas restantes, perdeu o jogo
CT 03	w	sem resultado
CT 04	h	sem resultado
CT 05	[null]	expected = NullPointerException.class
CT 06	“”	expected = IllegalArgumentException.class
CT 07	‘2’	expected = IllegalArgumentException.class
CT 08	“h”	resultado "h----"
CT 09	“h” “h”	expected = IllegalArgumentException.class
CT 10	[sem entrada]	"hello"

Casos de teste de Aceitação:

Em todos os casos de teste a palavra esperada é “hello”.

CT01: Testar com entrada do primeiro usuário define a palavra “ball” e, o segundo usuário adivinhar as letras ‘b’ ‘a’ ‘l’, esperando a vitória.

CT02: Testar com entrada do primeiro usuário define a palavra “cat” e, o segundo usuário adivinhar as letras ‘c’ ‘a’ ‘t’ , esperando a derrota.

CT03: Testar com entrada do primeiro usuário define a palavra “dog” e, o segundo usuário adivinhar as letras ‘d’ ‘e’ ‘l’ ‘c’ ‘a’ ‘s’ ‘v’, esperando a derrota com 0 tentativas restantes.

CT04: Testar com entrada do primeiro usuário define a palavra “dog” e, o segundo usuário adivinhar as letras ‘d’ ‘e’ ‘l’ ‘c’ ‘a’ ‘s’ ‘o’ ‘g’, esperando a vitória com 1 tentativas restantes.

CT05: Testar com entrada do primeiro usuário define a palavra “far” e, o segundo usuário adivinhar as letras ‘f’ ‘a’ e logo depois verifica seu progresso e insere a letra ‘r’ esperando a vitória.

Casos de teste	Instruções do usuário	Resultado do teste
CT 01	o primeiro usuário define a palavra “ball” e, o segundo usuário adivinha as letras “bal”	Resultado retorna vitória
CT 02	o primeiro usuário define a palavra “cat” e, o segundo usuário adivinha “car”	Resultado retorna derrota
CT 03	o primeiro usuário define a palavra “dog” e, o segundo usuário adivinha as letras “delcasv”	Derrota, tentativas restantes são 0
CT 04	o primeiro usuário define a palavra “dog” e, o segundo “delcasog”	Vitória, restante de tentativas 1
CT 05	o primeiro usuário define a palavra “far” segundo usuário adivinha “far”	Verifica que a barra de progresso está “fa”, Vitória.

Relatório de Teste:

JUnit

02/09/2022 19:44

Coverage Report > <empty package>

Current scope: [all classes](#) | <empty package name>

Coverage Summary for Package: <empty package>

Package	Class, %	Method, %	Line, %
<empty package>	33,3% (1/3)	57,1% (8/14)	47,5% (29/61)

Class	Class, %	Method, %	Line, %
Game	100% (1/1)	100% (8/8)	100% (29/29)
Hangman	0% (0/1)	0% (0/2)	0% (0/7)
Prompter	0% (0/1)	0% (0/4)	0% (0/25)

generated on 2022-09-02 19:41

```
        return progress;
    }

    public int getRemainingTries(){
        return (MAX_MISSES - mMisses.length());
    }

    public boolean isSolved(){
        return (getCurrentProgress().indexOf('-') == -1);
    }




    public String getAnswer(){
        return mAnswer;
    }
}
```


Game

```

1 public class Game {
2     public static final int MAX_MISSES=6;
3     private String mAnswer;
4     private String mHits;
5     private String mMisses;
6
7     public Game (String answer){
8         mAnswer = answer;
9         mHits="";
10        mMisses="";
11    }
12
13    private char validateGuess(char letter){
14        if(!Character.isLetter(letter)){
15            throw new IllegalArgumentException ("Input A Letter");
16        }
17
18        letter=Character.toLowerCase(letter);
19
20        if(mMisses.indexOf(letter) >=0 || mHits.indexOf(letter) >=0){
21            throw new IllegalArgumentException (letter + " Already been Guessed!\n");
22        }
23
24        return letter;
25    }
26
27    public boolean guess (String s){ //Taking a String
28        if(s.length() == 0){
29            throw new IllegalArgumentException ("No Letter Found");
30        }
31        return guess(s.charAt(0));
32    }
33
34    public boolean guess(char s){ //Taking a Character
35        s= validateGuess(s);
36        boolean isHit= mAnswer.indexOf(s) >=0;
37        if(isHit){
38            mHits+=s;
39        } else {
40            mMisses+=s;
41        }
42
43        return isHit;
44    }
45
46    public String getCurrentProgress(){
47        String progress="";
48
49        for(char letter : mAnswer.toCharArray()){
50            char display='-';
51            if(mHits.indexOf(letter)>=0){
52                display=letter;
53            }
54            progress+=display;
55        }
56
57        return progress;
58    }

```

 Game	100% (1/1)	100% (8/8)	100% (29...
 Hangman	0% (0/1)	0% (0/1)	0% (0/6)
 Prompter	0% (0/1)	0% (0/4)	0% (0/25)

Cucumber

```
Testing started at 20:27 ...
Connected to the target VM, address: '127.0.0.1:51255', transport: 'socket'
```

```
5 Scenarios (5 passed)
37 Steps (37 passed)
0m0,556s
```

```
Share your Cucumber Report with your team at https://reports.cucumber.io
Activate publishing with one of the following:

src/test/resources/cucumber.properties:      cucumber.publish.enabled=true
src/test/resources/junit-platform.properties: cucumber.publish.enabled=true
Environment variable:                        CUCUMBER_PUBLISH_ENABLED=true
JUnit:                                       @CucumberOptions(publish = true)

More information at https://cucumber.io/docs/cucumber/environment-variables/

Disable this message with one of the following:

src/test/resources/cucumber.properties:      cucumber.publish.quiet=true
src/test/resources/junit-platform.properties: cucumber.publish.quiet=true
```

```
Disconnected from the target VM, address: '127.0.0.1:51255', transport: 'socket'
```

```
Process finished with exit code 0
```

PIT test

```
=====
- Statistics
=====
>> Line Coverage: 31/66 (47%)
>> Generated 45 mutations Killed 25 (56%)
>> Mutations with no coverage 18. Test strength 93%
>> Ran 45 tests (1 tests per mutation)
```

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
3	47% <div><div>31/66</div></div>	56% <div><div>25/45</div></div>	93% <div><div>25/27</div></div>

Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage	Test Strength
default	3	47% <div><div>31/66</div></div>	56% <div><div>25/45</div></div>	93% <div><div>25/27</div></div>

Report generated by [PIT](#) 1.6.8

Pit Test Coverage Report

Package Summary

default

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
3	47% <div><div>31/66</div></div>	56% <div><div>25/45</div></div>	93% <div><div>25/27</div></div>

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
Game.java	100% <div><div>31/31</div></div>	93% <div><div>25/27</div></div>	93% <div><div>25/27</div></div>
Hangman.java	0% <div><div>0/8</div></div>	0% <div><div>0/5</div></div>	0% <div><div>0/0</div></div>
Prompter.java	0% <div><div>0/27</div></div>	0% <div><div>0/13</div></div>	0% <div><div>0/0</div></div>

Report generated by [PIT](#) 1.6.8

61	1. Replaced integer subtraction with addition → KILLED 2. replaced int return with 0 for Game::getRemainingTries → KILLED
65	1. replaced boolean return with false for Game::isSolved → KILLED 2. replaced boolean return with true for Game::isSolved → KILLED 3. negated conditional → KILLED 4. removed conditional - replaced equality check with false → KILLED
69	1. replaced return value with "" for Game::getAnswer → KILLED

Active mutators

- BOOLEAN_FALSE_RETURN
- BOOLEAN_TRUE_RETURN
- CONDITIONALS_BOUNDARY_MUTATOR
- EMPTY_RETURN_VALUES
- EXPERIMENTAL_SWITCH_MUTATOR
- INCREMENTS_MUTATOR
- INVERT_NEGS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR
- NULL_RETURN_VALUES
- PRIMITIVE_RETURN_VALS_MUTATOR
- REMOVE_CONDITIONALS_EQUAL_ELSE_MUTATOR
- VOID_METHOD_CALL_MUTATOR

Tests examined

- GameTest.test06(GameTest) (2 ms)
- GameTest.test01(GameTest) (8 ms)

Tests examined

- GameTest.test06(GameTest) (2 ms)
- GameTest.test01(GameTest) (8 ms)
- GameTest.test02(GameTest) (1 ms)
- GameTest.test03(GameTest) (1 ms)
- GameTest.test04(GameTest) (1 ms)
- GameTest.test07(GameTest) (0 ms)
- GameTest.test10(GameTest) (1 ms)
- GameTest.test08(GameTest) (0 ms)
- GameTest.test09(GameTest) (1 ms)

Report generated by PIT 1.6.8

GitHub: <https://github.com/ericjmachado/ts-2022>