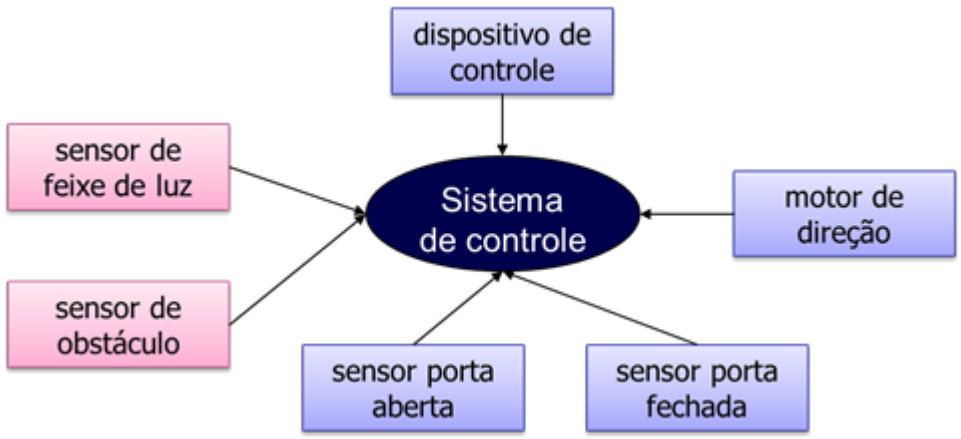
LINKS FERRAMENTAS

MÁQUINA DE ESTADOS
https://online.visual-paradigm.com/app/diagrams/#diagram:proj=0&type=StateMachineDiagram&width=11&height=8.5&unit=inch

GRAPH COVERAGE https://cs.gmu.edu:8443/offutt/coverage/GraphCoverage

Um sistema de controle de porta de garagem tem vários componentes, como mostra a figura: um motor, sensores que indicam se a porta está aberta ou fechada, um dispositivo de controle (possivelmente remoto) e dois sensores: de feixe de luz, na parte de baixo da porta, e um sensor de obstáculos, que atuam como dispositivos de segurança. Estes sensores operam somente quando a porta está fechando. No fechamento da porta, se o sensor do feixe de luz for interrompido (possivelmente por um animal de estimação) ou se a porta encontrar um obstáculo, a porta para imediatamente e reverte sua direção. Quando a porta está em movimento, seja fechando ou abrindo e um sinal do dispositivo de controle é recebido, a porta para. Se em seguida a porta recebe um sinal do controle, a porta volta a se mover no sentido em que estava antes de ser parada. Finalmente, se há sinal dos sensores que indicam se a porta está aberta ou fechada, a porta para.



Fonte: P. C. Jorgensen. The Craftof Model-BasedTesting. CRC Press, 2017, cap. 6.

Responda às questões a seguir:

1. (0,5pt) Identifique os eventos de entrada e as ações (saídas) produzidas pelo sistema.

Eventos de entrada:

- Sinal de controle
- Sensor de feixe de luz
- Sensor de obstáculo
- Sensor de porta aberta
- Sensor de porta fechada

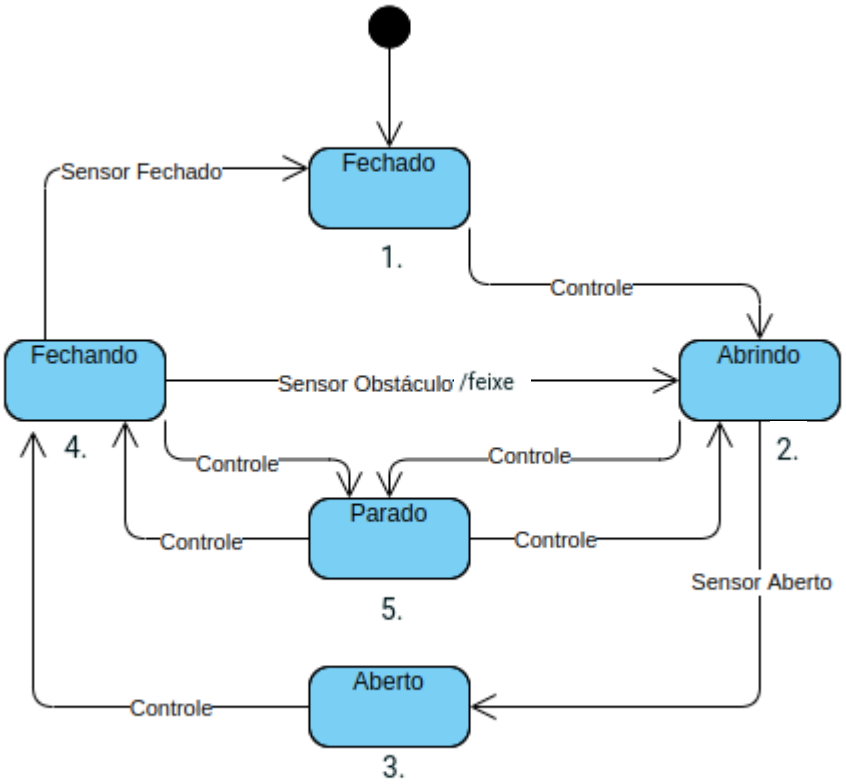
Ações:

- Para
- Volta a se mover no sentido em que estava antes de ser parada
- Reverte sua direção

2. (0,5) pt) Identifique os estados do Sistema de Controle.

1. Fechado
2. Abrindo
3. Aberto
4. Fechando
5. Parado

3. (0,5 pt) Construa o modelo de estados do sistema. Pode usar a ferramenta de sua preferência.



4. (0,5 pt) Usando a ferramenta GraphCoverage, crie os requisitos de teste para os diferentes critérios oferecidos.

Critério	Requisitos de Teste
Cobertura de estados	{[1],[2],[3],[5],[4]}
Cobertura de transições	{[1,2],[2,3],[2,5],[3,4],[4,1],[4,2],[4,5],[5,2],[5,4]}
Cobertura de pares de transições	{[1,2,3],[1,2,5],[2,3,4],[2,5,2],[2,5,4],[3,4,1],[3,4,2],[3,4,5],[4,1,2],[4,2,3],[4,2,5],[4,5,2],[4,5,4],[5,2,3],[5,2,5],[5,4,1],[5,4,2],[5,4,5]}
Cobertura de caminhos simples	{[1,2],[2,3],[2,5],[3,4],[4,1],[4,2],[4,5],[5,2],[5,4],[1,2,3],[1,2,5],[2,3,4],[2,5,2],[2,5,4],[3,4,1],[3,4,2],[3,4,5],[4,1,2],[4,2,3],[4,2,5],[4,5,2],[4,5,4],[5,2,3],[5,2,5],[5,4,1],[5,4,2],[5,4,5],[1,2,

	3,4],[1,2,5,4],[2,3,4,1],[2,3,4,2],[2,3,4,5],[2,5,4,1],[2,5,4,2],[3,4,1,2],[3,4,2,3],[3,4,2,5],[3,4,5,2],[4,1,2,3],[4,1,2,5],[4,2,3,4],[4,2,5,4],[4,5,2,3],[5,2,3,4],[5,4,1,2],[5,4,2,3],[5,4,2,5],[1,2,3,4,1],[1,2,3,4,5],[1,2,5,4,1],[2,3,4,1,2],[2,3,4,5,2],[2,5,4,1,2],[3,4,1,2,3],[3,4,1,2,5],[3,4,5,2,3],[4,1,2,3,4],[4,1,2,5,4],[4,5,2,3,4],[5,2,3,4,1],[5,2,3,4,5],[5,4,1,2,3],[5,4,1,2,5]}
Cobertura de caminhos primários	{ [2,5,4,1,2],[2,3,4,5,2],[3,4,1,2,3],[3,4,1,2,5],[2,3,4,1,2],[1,2,3,4,1],[1,2,3,4,5],[1,2,5,4,1],[3,4,5,2,3],[5,2,3,4,5],[5,4,1,2,3],[5,4,1,2,5],[5,2,3,4,1],[4,1,2,3,4],[4,1,2,5,4],[4,5,2,3,4],[2,5,4,2],[3,4,2,5],[3,4,2,3],[2,3,4,2],[5,4,2,3],[5,4,2,5],[4,2,3,4],[4,2,5,4],[2,5,2],[5,2,5],[5,4,5],[4,5,4]}

5. (0,5 pt) Usando a ferramenta GraphCoverage, crie os casos de teste para os diferentes critérios oferecidos.

Cobertura de estados

1 test path is needed for Node Coverage
[1,2,5,4,2,3]

Cobertura de transições

5 test paths are needed for Edge Coverage
[3,4,2,3]
[1,2,5,2,3]
[1,2,5,4,2,3]
[3,4,5,2,3]
[1,2,5,4,5,4,1]

Cobertura de pares de transições

8 test paths are needed for Edge-Pair Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,4,1]	[1,2,3], [2,3,4], [3,4,1]
[1,2,5,2,3]	[1,2,5], [2,5,2], [5,2,3]
[3,4,2,3]	[3,4,2], [4,2,3]
[3,4,5,4,1]	[3,4,5], [4,5,4], [5,4,1]
[1,2,5,4,1,2,3]	[1,2,3], [1,2,5], [2,5,4], [4,1,2], [5,4,1]
[1,2,5,4,2,5,4,1]	[1,2,5], [2,5,4], [4,2,5], [5,4,1], [5,4,2]
[1,2,5,4,5,2,3]	[1,2,5], [2,5,4], [4,5,2], [5,2,3], [5,4,5]
[1,2,5,2,5,4,1]	[1,2,5], [2,5,2], [2,5,4], [5,2,5], [5,4,1]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,3,4,1]	None
[1,2,5,2,3]	[1,2,3]
[3,4,2,3]	None
[3,4,5,4,1]	[3,4,1]
[1,2,5,4,1,2,3]	None
[1,2,5,4,2,5,4,1]	None
[1,2,5,4,5,2,3]	[1,2,5], [2,5,2]
[1,2,5,2,5,4,1]	[1,2,5], [2,5,4]

Infeasible Edge-Pairs are:
None

Cobertura de caminhos primários

15 test paths are needed for Prime Path Coverage

Test Paths	Test Requirements that are toured by test paths directly
[1,2,3,4,5,2,3]	[2,3,4,5,2], [1,2,3,4,5], [3,4,5,2,3]
[3,4,1,2,5,4,1]	[3,4,1,2,5], [1,2,5,4,1], [4,1,2,5,4]
[1,2,3,4,1,2,3]	[3,4,1,2,3], [2,3,4,1,2], [1,2,3,4,1]
[1,2,3,4,5,4,1]	[1,2,3,4,5], [4,5,4]
[1,2,5,2,3,4,5,4,1]	[5,2,3,4,5], [2,5,2], [4,5,4]
[1,2,5,4,1,2,5,4,1]	[2,5,4,1,2], [1,2,5,4,1], [5,4,1,2,5], [4,1,2,5,4]
[1,2,5,2,3,4,1]	[5,2,3,4,1], [2,5,2]
[1,2,5,4,1,2,3,4,1]	[2,5,4,1,2], [1,2,3,4,1], [1,2,5,4,1], [5,4,1,2,3], [4,1,2,3,4]
[1,2,5,4,5,2,3,4,1]	[5,2,3,4,1], [4,5,2,3,4], [5,4,5]
[3,4,2,5,4,1]	[3,4,2,5], [4,2,5,4]
[1,2,3,4,2,3]	[3,4,2,3], [2,3,4,2]
[1,2,5,4,2,5,4,1]	[2,5,4,2], [5,4,2,5], [4,2,5,4]
[1,2,5,4,2,3,4,1]	[2,5,4,2], [5,4,2,3], [4,2,3,4]
[1,2,5,2,5,4,1]	[2,5,2], [5,2,5]
[1,2,5,4,5,4,1]	[5,4,5], [4,5,4]

Test Paths	Test Requirements that are toured by test paths with sidetrips
[1,2,3,4,5,2,3]	None
[3,4,1,2,5,4,1]	None
[1,2,3,4,1,2,3]	None
[1,2,3,4,5,4,1]	[1,2,3,4,1]
[1,2,5,2,3,4,5,4,1]	[1,2,3,4,5], [1,2,5,4,1], [5,2,3,4,1]
[1,2,5,4,1,2,5,4,1]	[1,2,5,4,1]
[1,2,5,2,3,4,1]	[1,2,3,4,1]
[1,2,5,4,1,2,3,4,1]	[1,2,3,4,1], [1,2,5,4,1]
[1,2,5,4,5,2,3,4,1]	[1,2,5,4,1], [2,5,2]
[3,4,2,5,4,1]	None
[1,2,3,4,2,3]	None
[1,2,5,4,2,5,4,1]	[1,2,5,4,1]
[1,2,5,4,2,3,4,1]	[1,2,3,4,1], [1,2,5,4,1]
[1,2,5,2,5,4,1]	[1,2,5,4,1]
[1,2,5,4,5,4,1]	[1,2,5,4,1]

Infeasible prime paths are:
None

6. (2) Qual dos critérios você usaria se tivesse pouco tempo para os testes? Qual dos critérios você usaria se quiser maior potencial para revelar defeitos? Justifique suas respostas.

Se tivesse pouco tempo para os testes, usaria os critérios de testes que incluem as transições do estado 4 para o estado 2, do estado 1 para 2 e do estado 3 para 4. Pois acredito que garantir a segurança dos animais de estimação e o funcionamento dos sinais de controle para abrir e fechar a porta são os mais importantes.

Para aumentar o potencial de revelar defeitos, entretanto, usaria os critérios de teste que incluem as transições entre os estados 4, 5 e 2, pois é onde reside a maior complexidade da MEF e maior probabilidade de surgirem defeitos.

Considere a seguinte interface IntList, que implementa uma lista com elementos do tipo int. Esta classe tem os seguintes métodos:

Métodos	Descrição
IntList ( )	constrói lista vazia
IntList(v: int [ ])	constrói lista com elementos do vetor de inteiros
void add(int index, int element)	insere element na posição especificada por index . Lança IndexOutOfBoundsException se index <0 ou index > size( )
boolean addAll(v: int[ ])	insere os elementos de v ao final da lista; retorna true se inseriu, false em caso contrário. Lança IndexOutOfBoundsException se tamanho de v > size( )
void clear( )	limpa a lista, removendo todos os seus elementos
boolean contains(int element)	retorna true se IntList contém elemento
int get(int index)	retorna o elemento que está na posição index
int indexOf(int elem)	busca a primeira ocorrência de elem na lista; retorna -1 se elem não está na lista
boolean isEmpty( )	retorna true se a lista está vazia
int lastIndexOf(int elem)	retorna a posição da última ocorrência de elem na lista; retorna -1 caso elem não esteja na lista

int remove (int index)	retorna o elemento removido da posição index Lança IndexOutOfBoundsException se index <0 ou index > size( )
int size( )	retorna o nº atual de elementos da lista

As questões a seguir são referentes a esta classe.
1. (1pt) Identifique os eventos e as saídas esperadas.

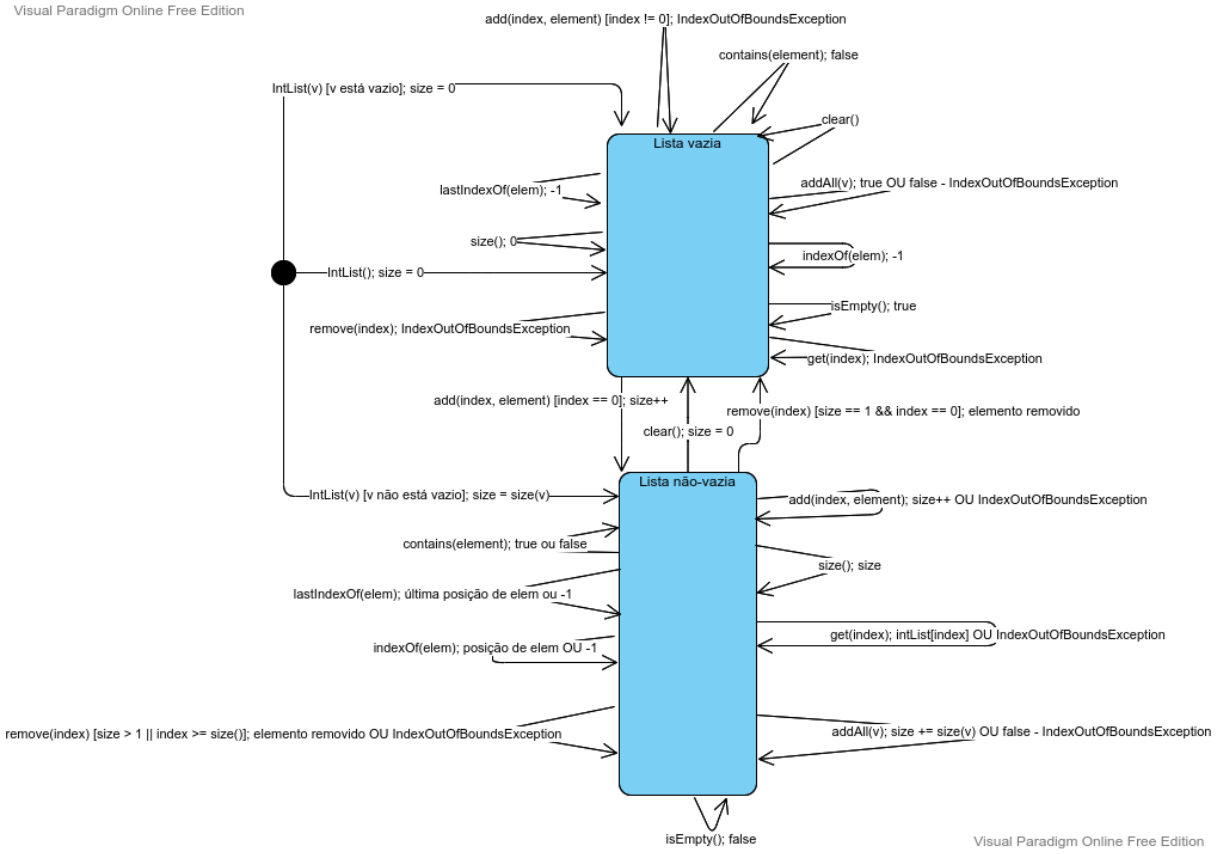
- IntList() -> lista vazia
- IntList(v: int [ ]) -> lista com elementos do vetor de inteiros
- add(int index, int element) -> insere element na posição index; IndexOutOfBoundsException
- addAll(v: int[ ]) -> insere os elementos de v ao final da lista e retorna true; retorna false; IndexOutOfBoundsException se tamanho de v > size( )
- clear( ) -> limpa a lista
- contains(int element) -> true; false
- get(int index) -> elemento na posição index; IndexOutOfBoundsException (como o comportamento não foi explicitado na especificação, assumimos um comportamento coerente com o restante das especificações)
- indexOf(int elem) -> posição de elem; -1
- isEmpty( ) -> true; false
- lastIndexOf(int elem) -> última posição de elem; -1
- remove (int index) -> elemento removido da posição index; IndexOutOfBoundsException
- size( ) -> número atual de elementos na lista

2. (1,5pt) Identifique os estados de IntList. Quais invariantes caracterizam os estados? Indique as variáveis de contexto criadas para caracterizar estas invariantes.

1. Lista vazia
2. Lista não-vazia

   Os estados são caracterizados pelo tamanho da lista, representado pela variável de contexto size.

3. (1 pt) Construa o modelo de estados. Pode usar ferramenta de sua preferência.



Link para edição: https://online.visual-paradigm.com/share.jsp?id=313935343533352d31

4. (2 pt) Usando a ferramenta GraphCoverage, crie os requisitos de teste para os diferentes critérios oferecidos. Identifique se há requisitos infactíveis. Justifique.

Critério	Requisitos de Teste
Cobertura de estados	{[1],[2]}
Cobertura de transições	{[1,1],[2,2],[1,2],[2,1]}
Cobertura de pares de transições	{[1,1,1],[1,1,2],[2,2,2],[2,2,1],[1,2,2],[1,2,1],[2,1,1],[2,1,2]}
Cobertura de caminhos simples	{[1,1],[2,2],[1,2],[2,1],[1,2,1],[2,1,2]}
Cobertura de caminhos primários	{[1,2,1],[2,1,2],[2,2],[1,1]}

Não há requisitos infactíveis, pois há apenas 2 estados e todas as transições são possíveis (inclusive cíclicas).