5) Suponha que uma lista inicialmente vazia S tenha executado um total de 25 operações push, 12 operações top e 10 operações pop, 3 das quais geraram StackEmptyExceptions, que foram capturadas e ignoradas. Qual é o tamanho corrente de S?

Operações	Saída	Conteúdo
pop()	"erro"	()
pop()	"erro"	()
top()	"erro"	()
push(3)	-	(3)
top()	3	(3)
push(5)	-	(5, 3)
top()	5	(5, 3)
push(2)	-	(2, 5, 3)
top()	2	(2, 5, 3)
push(8)	-	(8, 2, 5, 3)
top()	8	(8, 2, 5, 3)
push(4)	-	(4, 8, 2, 5, 3)
pop()	4	(8, 2, 5, 3)
pop()	8	(2, 5, 3)
pop()	2	(5, 3)
push(7)	-	(7, 5, 3)
top()	7	(7, 5, 3)
pop()	7	(5, 3)
pop()	5	(3)
push(2)	-	(2, 3)
push(5)	-	(5, 2, 3)
push(3)	-	(3, 5, 2, 3)
push(7)	-	(7, 3, 5, 2, 3)
push(1)	-	(1, 7, 3, 5, 2, 3)
push(8)	-	(8, 1, 7, 3, 5, 2, 3)
push(9)	-	(9, 8, 1, 7, 3, 5, 2, 3)
top()	9	(9, 8, 1, 7, 3, 5, 2, 3)
pop()	9	(8, 1, 7, 3, 5, 2, 3)
top()	8	(8, 1, 7, 3, 5, 2, 3)
pop()	8	(1, 7, 3, 5, 2, 3)
pop()	1	(7, 3, 5, 2, 3)
top()	7	(7, 3, 5, 2, 3)
psuh(4)	-	(4, 7, 3, 5, 2, 3)
push(6)	-	(6, 4, 7, 3, 5, 2, 3)
top()	6	(6, 4, 7, 3, 5, 2, 3)
push(2)	-	(2, 6, 4, 7, 3, 5, 2, 3)
push(5)	-	(5, 2, 6, 4, 7, 3, 5, 2, 3)
top()	5	(5, 2, 6, 4, 7, 3, 5, 2, 3)
push(9)	-	(9, 5, 2, 6, 4, 7, 3, 5, 2, 3)
push(7)	-	(7, 9, 5, 2, 6, 4, 7, 3, 5, 2, 3)
top()	7	(7, 9, 5, 2, 6, 4, 7, 3, 5, 2, 3)
push(9)	-	(9, 7, 9, 5, 2, 6, 4, 7, 3, 5, 2, 3)
push(6)	-	(6, 9, 7, 9, 5, 2, 6, 4, 7, 3, 5, 2, 3)
push(56)	-	(56, 6, 9, 7, 9, 5, 2, 6, 4, 7, 3, 5, 2, 3)
push(89)	-	(89, 56, 6, 9, 7, 9, 5, 2, 6, 4, 7, 3, 5, 2, 3)
push(24)	-	(24, 89, 56, 6, 9, 7, 9, 5, 2, 6, 4, 7, 3, 5, 2, 3)
push(12)	-	(12, 24, 89, 56, 6, 9, 7, 9, 5, 2, 6, 4, 7, 3, 5, 2, 3)

O tamanho corrente de S é de 17.

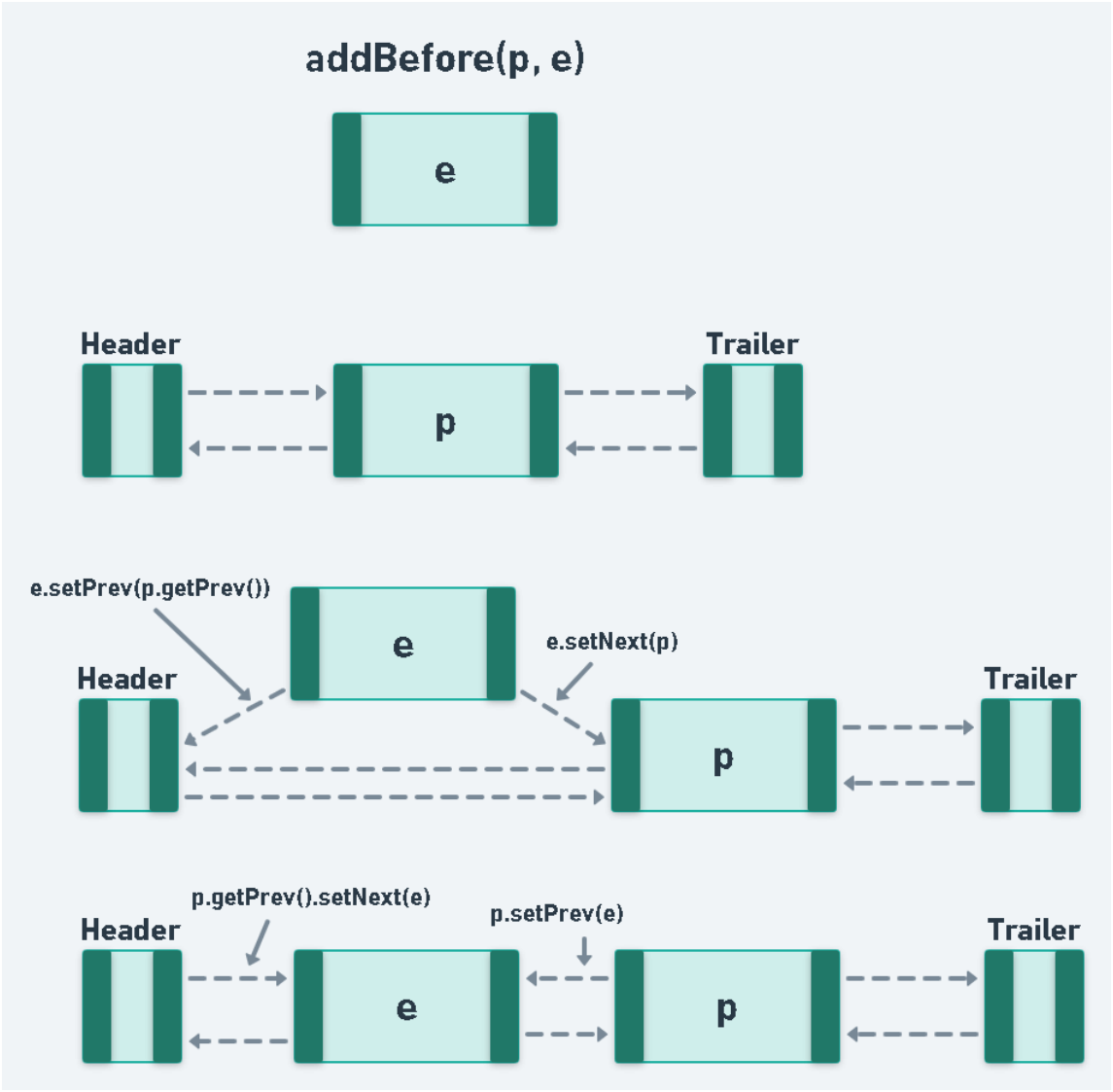
6) Se implementarmos a pilha S do problema anterior usando um arranjo, então qual será o valor corrente da variável de instância top?

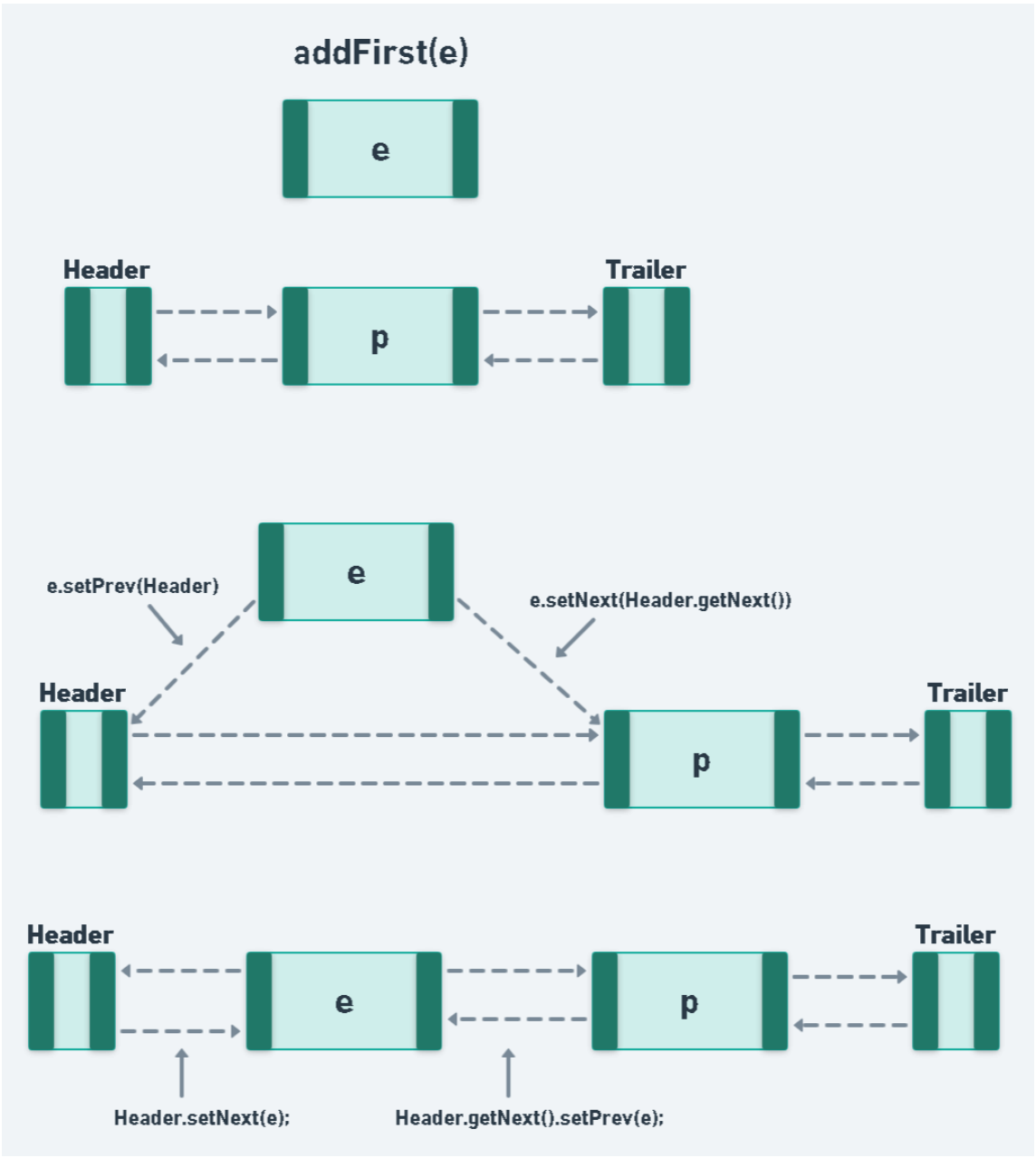
O valor corrente da variável de instância top será o último elemento a ser adicionando com o método push, ou seja, como visto na questão anterior o valor é 12.

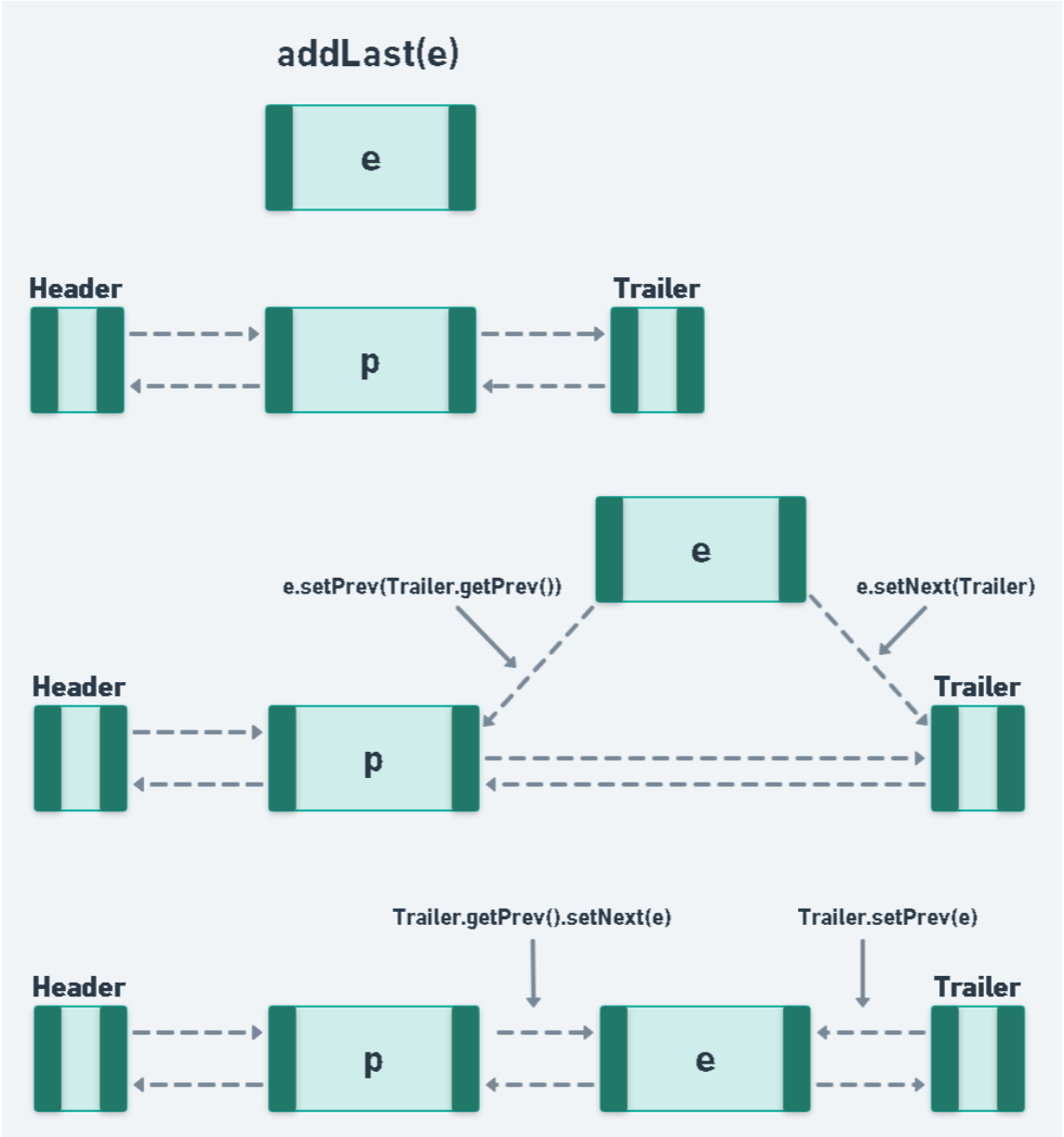
7) Descreva a saída resultante da seguinte série de operações de pilha: push(5), push(3), pop( ),push(2), push(8), pop( ), pop( ), push(9), push(1), pop( ), push(7), push(6), pop(), pop(), push(4), pop(), pop( ).

Operações	Saída	Conteúdo
push(5)	-	(5)
push(3)	-	(3, 5)
pop()	3	(5)
push(2)	-	(2, 5)
push(8)	-	(8, 2, 5)
pop()	8	(2, 5)
pop()	2	(5)
push(9)	-	(9, 5)
push(1)	-	(1, 9, 5)
pop()	1	(9, 5)
push(7)	-	(7, 9, 5)
push(6)	-	(6, 7, 9, 5)
pop()	6	(7, 9, 5)
pop()	7	(9, 5)
push(4)	-	(4, 9, 5)
pop()	4	(9, 5)
pop()	9	(5)

10) Desenhe figuras demonstrando cada um dos passos principais dos métodos addBefore(p, e), addFirst(e) e addLast(e) do TAD lista de nodos.



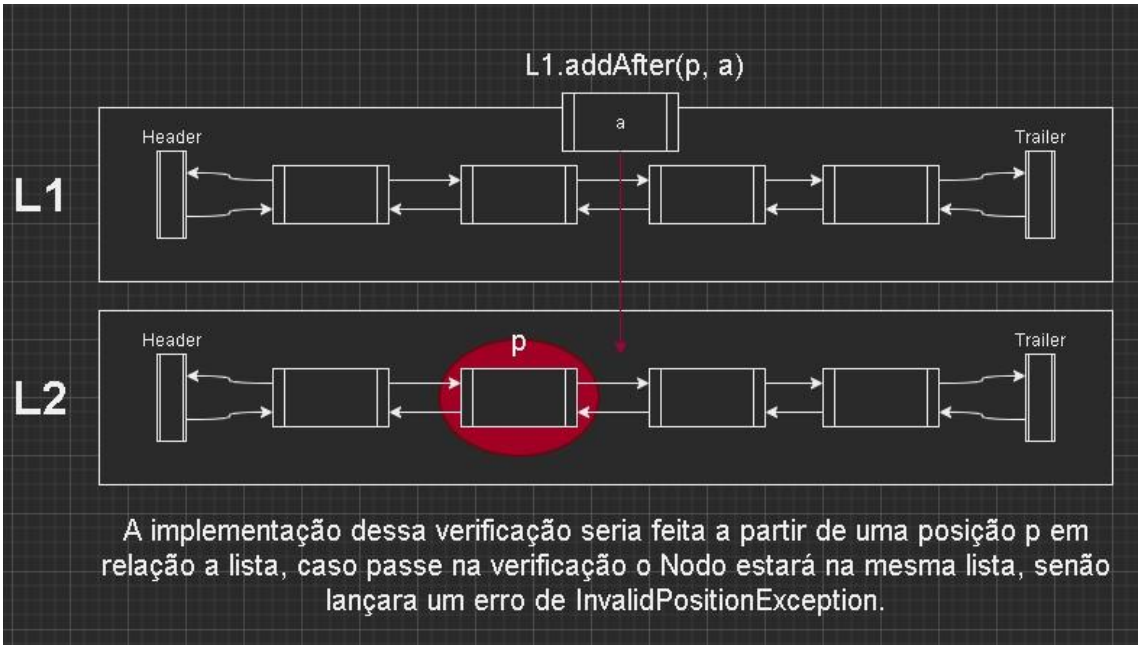

addFirst(e)
e
Header
p
Trailer
e.setPrev(Header)
e
e.setNext(Header.getNext())
Header
p
Trailer
Header
e
p
Trailer
Header.setNext(e);
Header.getNext().setPrev(e);



13) A implementação de NodePositionList não faz verificações de erro para testar se uma dada posição p é realmente membro dessa lista em particular.

a. Por exemplo, se p é uma posição da lista S, a execução T.addAfter(p,e) deveria lançar a exceção InvalidPositionException pois p não é uma posição de T.

b. Descreva como alterar a implementação de NodePositionList de uma forma eficiente que impeça esses maus usos.



A implementação dessa verificação seria feita a partir de uma posição p em relação a lista, caso passe na verificação o Nodo estará na mesma lista, senão lançara um erro de InvalidPositionException.