


5. 스택

개요

- 채소 가게의 과일 상자
 - 꼭대기가 바나나 상자라면
 - 바나나는 다른 상자들을 건드리지 않고 꺼낼 수 있음
 - 오렌지는 그 위에 있는 상자들을 들어내야 꺼낼 수 있음

5.1 스택 ADT

- 스택(stack)
 - 후입선출(LIFO: last-in-first-out) 프로토콜을 구현하는 자료 구조
 - 접근 가능한 유일한 객체는 가장 최근에 삽입된 객체
 - 4개 연산: Peek, Pop, Push, Size
 - 삽입 / 삭제가 일어나는 쪽을 스택의 top이라고 부름
- 연산  (스택 top의 원소를 제거하지 않고 반환)
 1. Peek: 스택이 공백이 아니면, 톱의 원소를 리턴한다.
 2. Pop: 스택이 공백이 아니면, 톱의 원소를 삭제해서 리턴한다. (스택의 top에 있는 원소를 제거하고 반환)
 3. Push: 주어진 원소를 스택의 톱에 추가한다. (스택에 새로운 원소를 삽입)
 4. Size: 스택에 있는 원소의 수를 리턴한다.

Stack 인터페이스

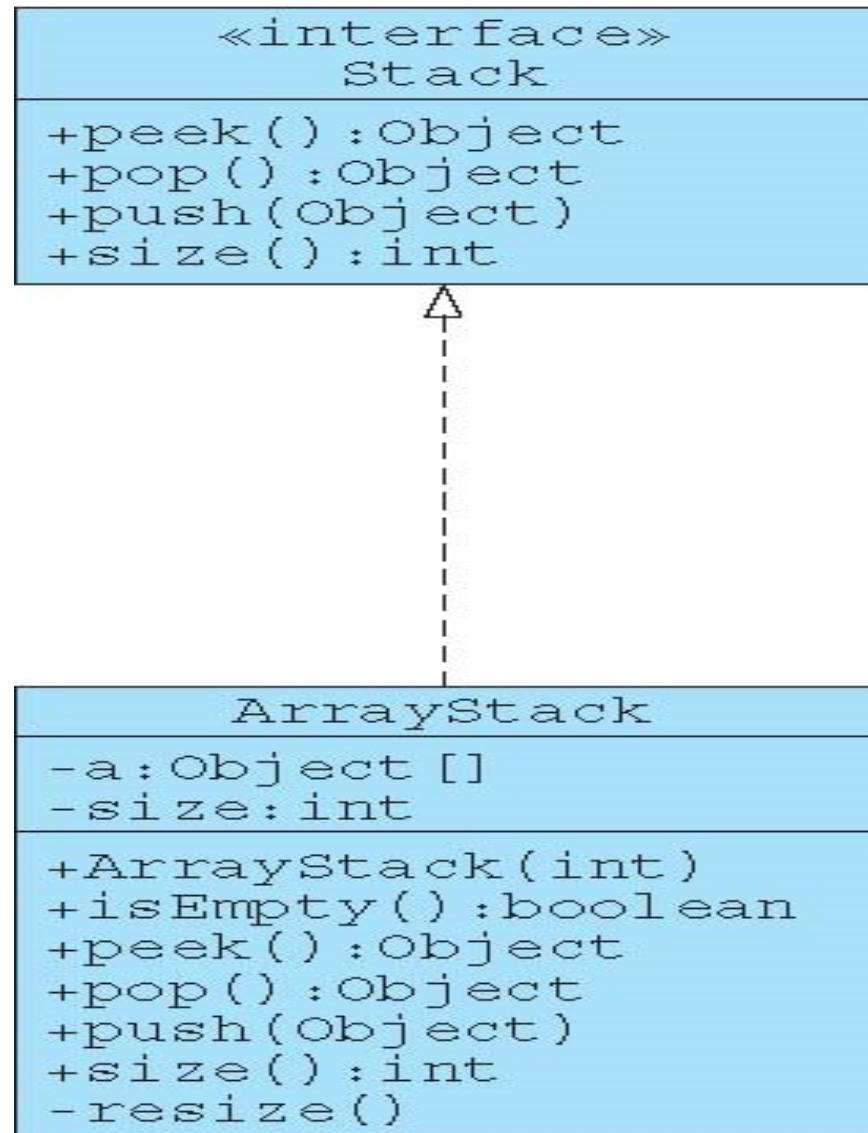
<code>«interface» Stack</code>
<code>+peek() : Object +pop() : Object +push(Object) +size() : int</code>

```
public interface Stack {  
    public Object peek();  
    public Object pop();  
    public void push(Object object);  
    public int size();  
}
```

5.2 배열 구현

- 리스팅 5.2: ArrayStack 클래스
 - 스택의 원소를 저장하기 위해 배열 `a[]` 사용
 - 스택에 있는 원소의 수를 세기 위해 정수 `size` 사용
 - 4개 연산(Peek, Pop, Push, Size) 외에 `isEmpty()`와 `resize()` 메소드 포함
 - `resize()`는 배열이 꽉 찼을 때 `push()`에 의해서 호출

스택 인터페이스의 구현



LISTING 5.2: ArrayStack 클래스

```
1 public class ArrayStack implements Stack {
2     private Object[] a;
3     private int size;

5     public ArrayStack(int capacity) {
6         a = new Object[capacity];
7     }

9     public boolean isEmpty() {
10         return (size == 0);
11     }

13    public Object peek() {
14        if (size == 0) throw new IllegalStateException("stack is
15            empty");
16        return a[size-1];
17    }
```

```
18  public Object pop() {
19      if (size == 0) throw new IllegalStateException("stack is
        empty");
20      Object object = a[--size];
21      a[size] = null;
22      return object;
23  }
25  public void push(Object object) {
26      if (size == a.length) resize();
27      a[size++] = object;
28  }

30  public int size() {
31      return size;
32  }
```



```
34  private void resize() {  
35      Object[] aa = a;  
36      a = new Object[2*aa.length];  
37      System.arraycopy(aa, 0, a, 0, size);  
38  }  
39 }
```

ArrayStack 클래스의 테스트

```
1 public class TestArrayStack {
2     public static void main(String[] args) {
3         Stack crates = new ArrayStack(4);
4         crates.push("CARROTS");
5         crates.push("ORANGES");
6         crates.push("RAISINS");
7         crates.push("PICKLES");
8         crates.push("BANANAS");
9         System.out.println("crates.size(): " + crates.size() + " " + crates.peek(): "
10                                + crates.peek());
11         System.out.println("crates.pop(): " + crates.pop());
12         System.out.println("crates.pop(): " + crates.pop());
13         System.out.println("crates.pop(): " + crates.pop());
14         System.out.println("crates.size(): " + crates.size() + " " + crates.peek(): "
15                                + crates.peek());
```

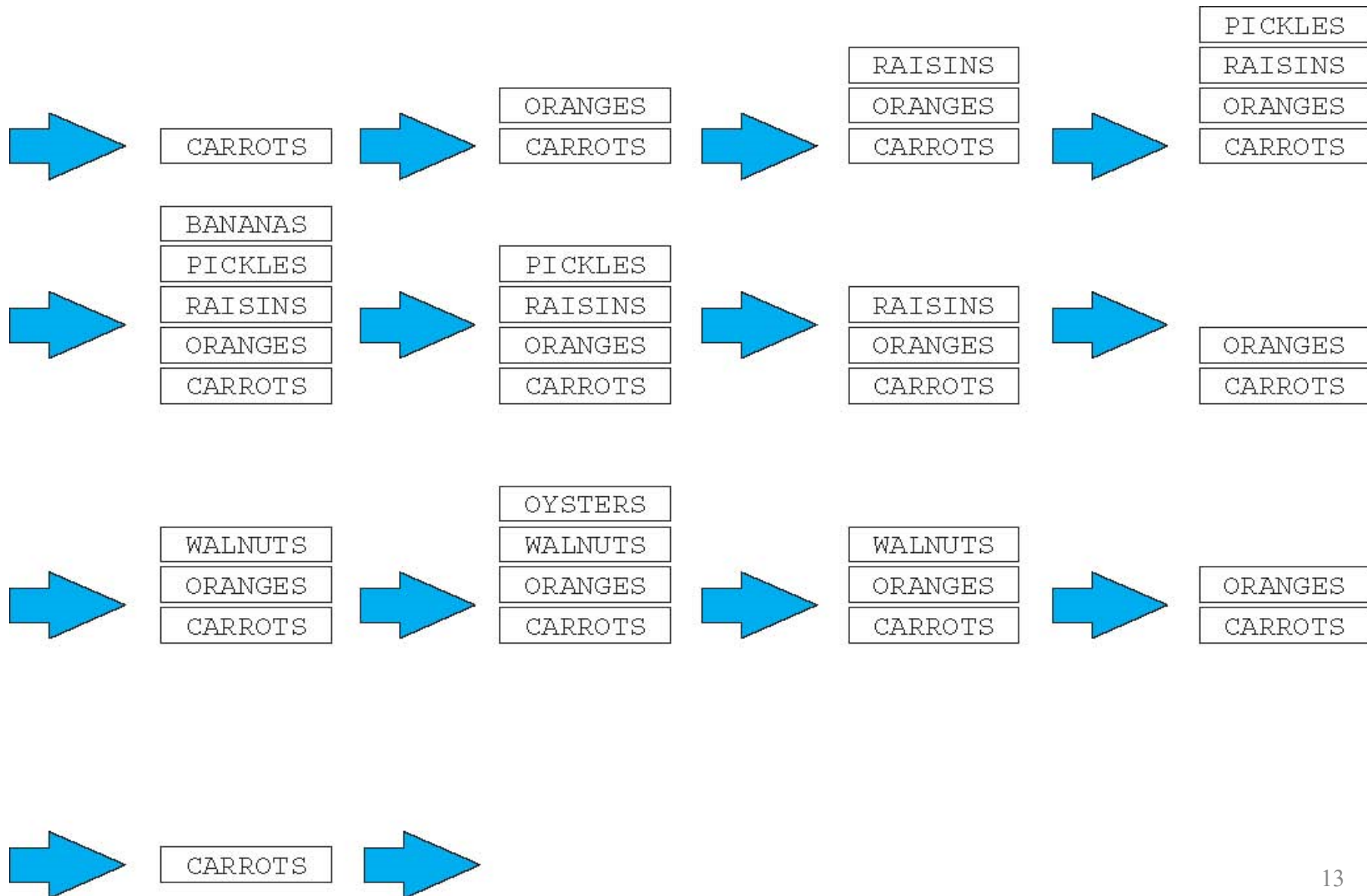
ArrayStack 클래스의 테스트

```
16     cratecrates.push("WALNUTS");
17     cratecrates.push("OYSTERS");
18     System.out.println("crates.size(): " + crates.size() +
19         "\tcrates.peek(): " + crates.peek());
20     System.out.println("crates.pop(): " + crates.pop());
21     System.out.println("crates.pop(): " + crates.pop());
22     System.out.println("crates.pop(): " + crates.pop());
23     System.out.println("crates.pop(): " + crates.pop());
24     System.out.println("crates.pop(): " + crates.pop());
25 }
26 }
```

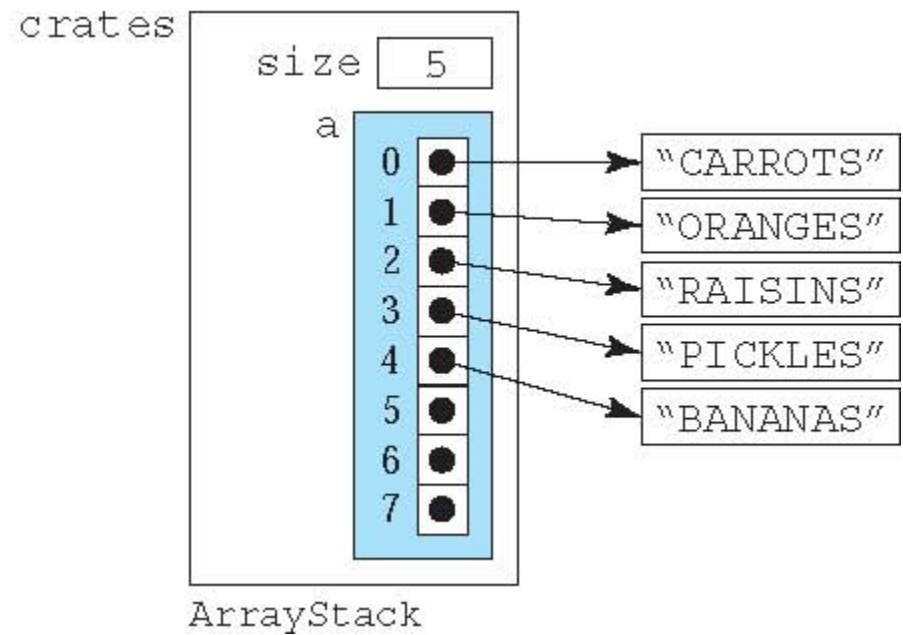
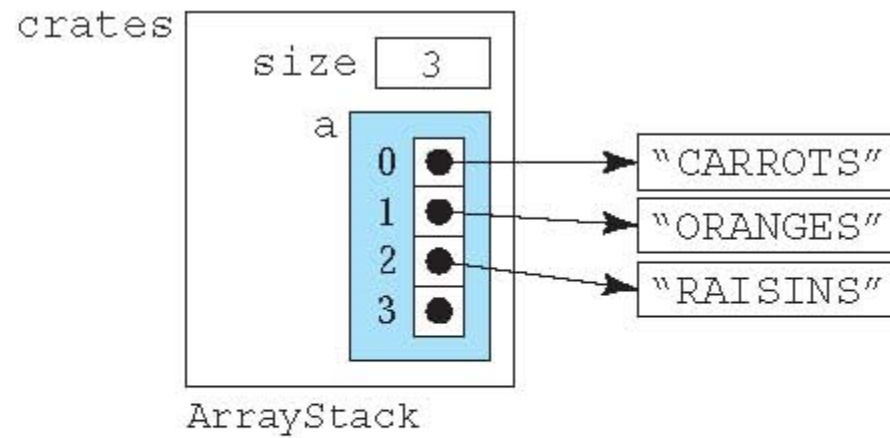
- 출력 결과

```
crates.size(): 5          crates.peek(): BANANAS
crates.pop(): BANANAS
crates.pop(): PICKLES
crates.pop(): RAISINS
crates.size(): 2          crates.peek(): ORANGES
crates.size(): 4          crates.peek(): OYSTERS
crates.pop(): OYSTERS
crates.pop(): WALNUTS
crates.pop(): ORANGES
crates.pop(): CARROTS
java.lang.IllegalStateException: stack is empty
Exception in thread main
```

리스팅 5.3의 프로그램에 대한 클라이언트의 관점

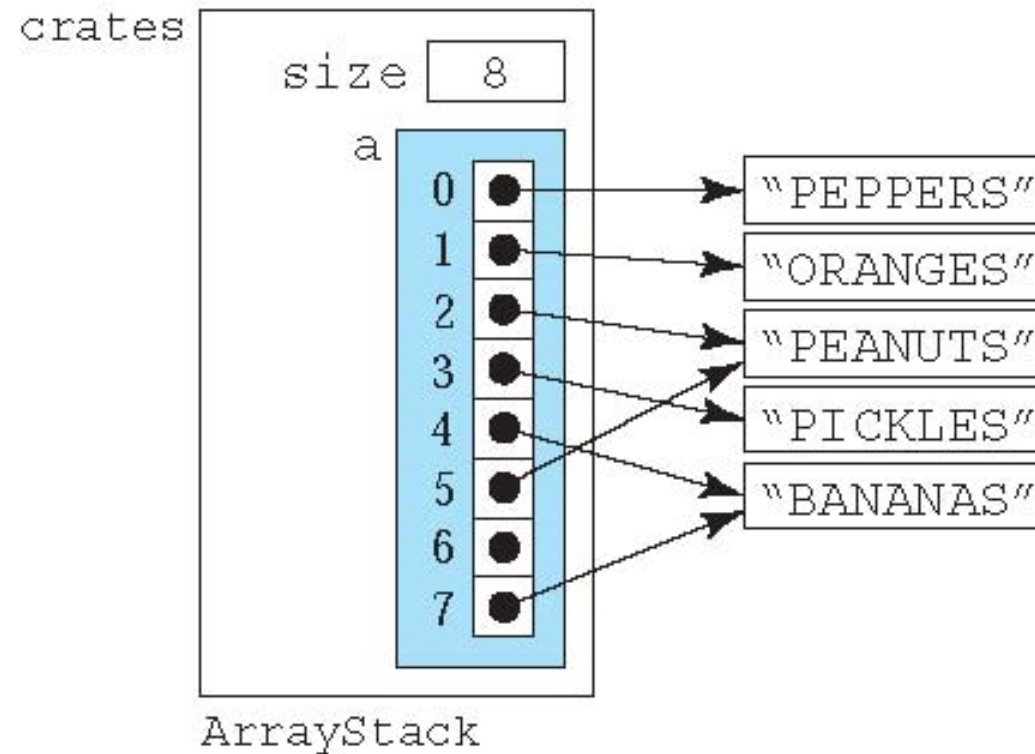


리스트 5.3의 프로그램에 대한 구현자의 관점



이 구현에서 가능한 이상 현상

- 스택의 여러 원소가 동일 객체를 참조
- 일부 참조가 null이 되는 것 허용할 수 있다



5.3 응용: 후위식의 평가

- 중위 표기 (*infix notation*)
 - 일상적인 산술식의 표기
 - 예: $(8 - 3) * (5 + 6)$
- 전위표기 (*prefix notation*)
 - 폴란드 논리학자인 얀 우카시에비치가 1920년대 명제 논리를 단순하게 하려고 발명
 - 폴란드 표기(Polish notation)라고도 함
 - 예: $* - 8 3 + 5 6$
- 후위 표기 (*postfix notation*)
 - 연산자가 항상 피연산자 뒤에 나옴
 - 역 폴란드 표기(reverse Polish notation)라고도 함
 - 예: $8 3 - 5 6 + *$

후위식 평가 프로그램: 리스팅 5.4

- LISTING 5.4: An RPN Calculator

java RPN 7 2 A 5 8 4 D S M

```
1 public class RPN {
2     public RPN(String[] args) {
3         Stack stack = new ArrayStack(args.length);
4         for (int i = 0; i < args.length; i++) {
5             String input = args[i];
6             if (isAnOperator(input)) {
7                 double y = Double.parseDouble((String)stack.pop());
8                 double x = Double.parseDouble((String)stack.pop());
9                 double z = evaluate(x, y, input);
10                stack.push("" + z);
11            }
12            else stack.push(input);
13        }
14    }
```

```
16  private boolean isAnOperator(String s) {
17      return (s.length() == 1 && "ASMD".indexOf(s) >= 0);
18  }
20  private double evaluate(double x, double y, String op) {
21      double z = 0;
22      if (op.equals("A")) z = x + y;
23      else if (op.equals("S")) z = x - y;
24      else if (op.equals("M")) z = x * y;
25      else z = x / y;
26      System.out.println(x + " " + op + " " + y + " = " + z);
27      return z;
28  }
30  public static void main(String[] args) {
31      new RPN(args);
32  }
33 }
```

- 출력 결과

7.0 A 2.0 = 9.0

8.0 D 4.0 = 2.0

5.0 S 2.0 = 3.0

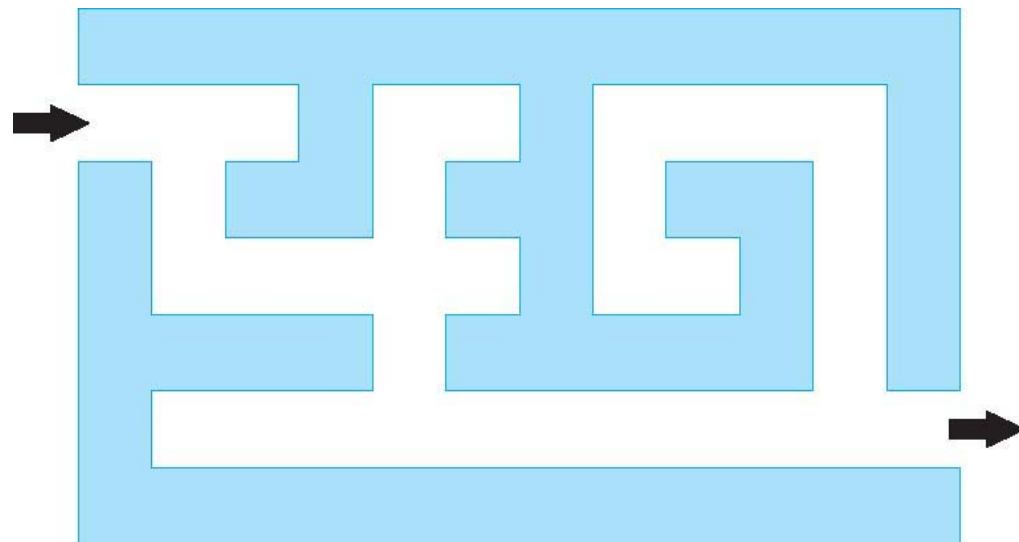
9.0 M 3.0 = 27.0

Transform Infix to Postfix

- 보충 강의자료(ch05_1.pdf) 참고...

5.4 사례 연구: 미로 풀기

- 문제: 왼쪽 상단의 입구에서 오른쪽 하단의 출구까지의 미로의 경로를 발견하는 것



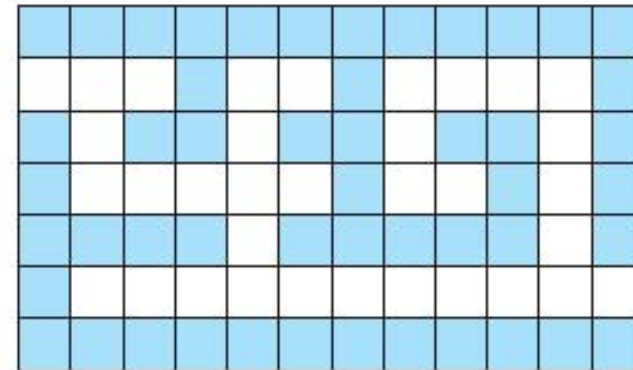
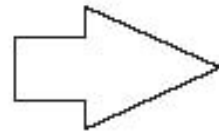
텍스트 파일에서 미로를 적재

텍스트 파일에서 미로를 적재

- 0은 통로의 일부, 1은 벽의 일부를 표현

Maze.txt

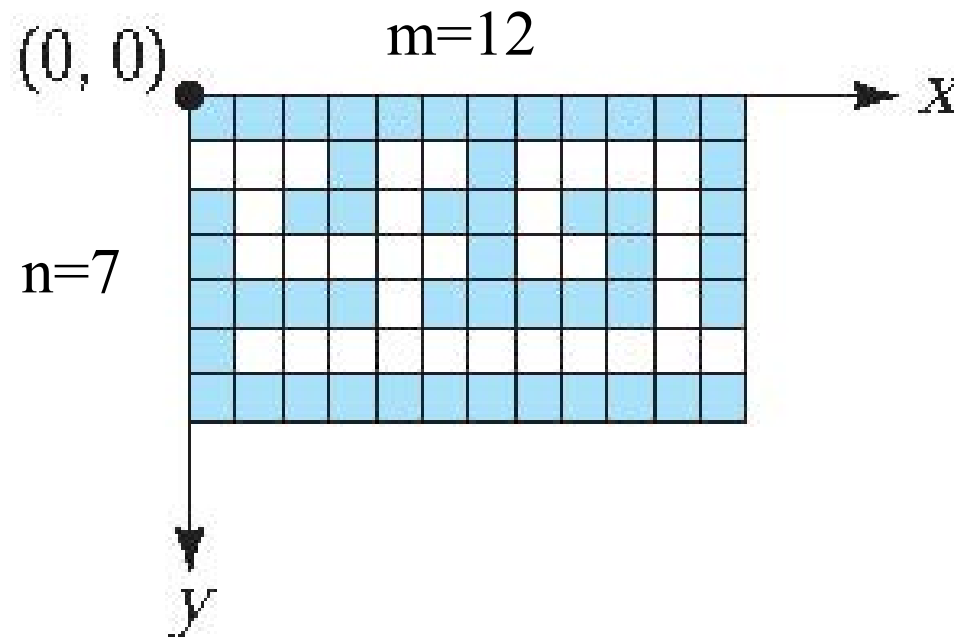
```
7
12
11111111111111
000100100001
101101101101
100000100101
111101111101
100000000000
11111111111111
```



좌표

좌표의 표현

- 왼쪽 상단 코너가 원점 (0,0)

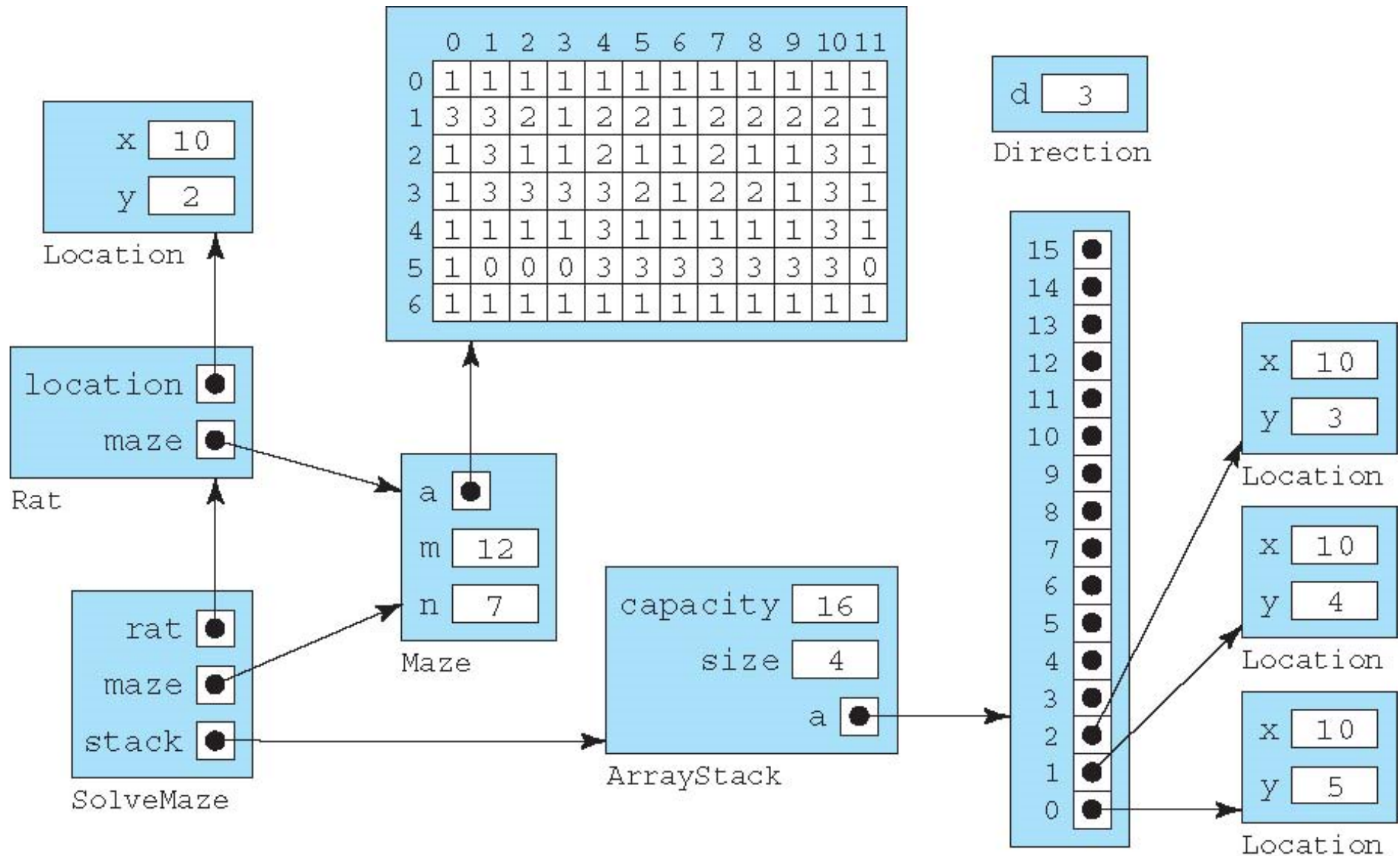


미로문제를 푸는 백트래킹 알고리즘

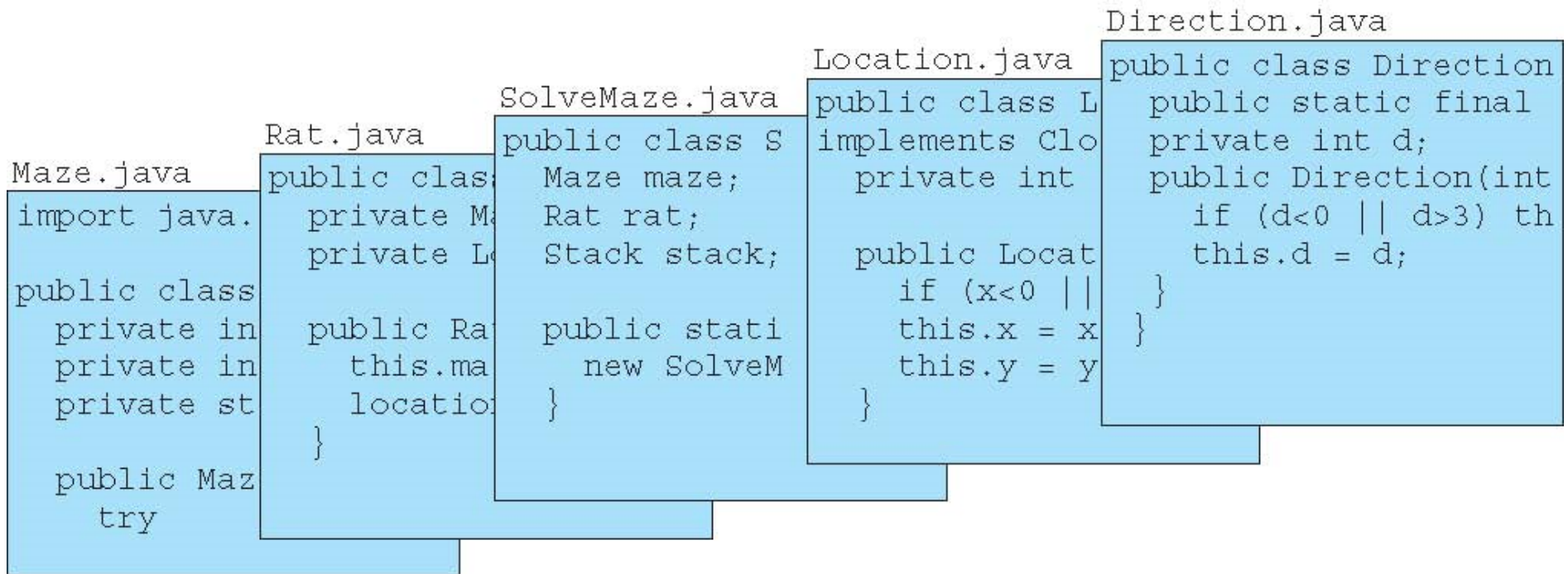
- 만약 쥐가 네 방향(동,서,남,북)중의 하나로 이동할 수 있으면, 현재 위치를 스택에 저장하고 그 방향의 이웃한 위치로 이동한다.
- 그렇지 않을 경우, 스택이 공백이면, 해가 없음을 보고하고 종료한다.
- 그렇지 않으면, 미로의 현재 위치를 "시도함(tried)"으로 표시하고, 스택에서 마지막 위치를 꺼내, 쥐를 그 위치로 다시 이동시킨다.

미로 솔루션에 대한 자료 구조

OPEN = 0, WALL = 1, TRIED = 2, PATH = 3



미로 솔루션을 위한 클래스들



미로 문제에 대한 솔루션의 추적

[illegible][illegible][illegible][illegible][illegible][illegible][illegible][illegible][illegible]

• • •

[illegible][illegible]

미로 풀기

- LISTING 5.5: Solving a Maze

```
1 public class SolveMaze {
2     Maze maze;
3     Rat rat;
4     Stack stack;
5
6     public static void main(String[] args) {
7         new SolveMaze(args[0]);
8     }
9
10    public SolveMaze(String file) {
11        maze = new Maze(file);
12        rat = new Rat(maze);
13        stack = new ArrayStack();
14        maze.print();
15        while (!rat.isOut()) {
16            Location currentLocation = rat.getLocation();
17            // see Programming Problem 5.23 on page 173
18        }
19    }
20 }
```

Maze 클래스

- LISTING 5.6: The Maze Class

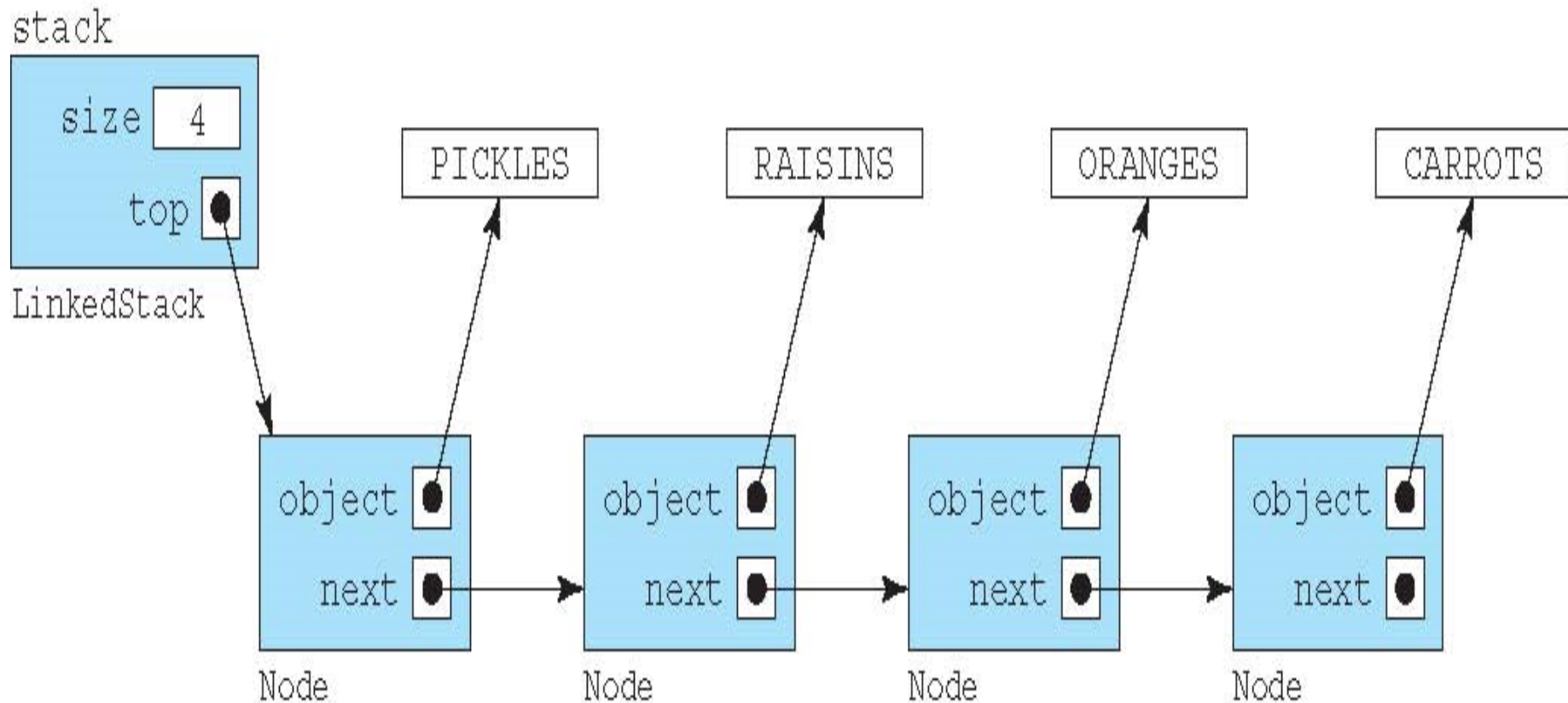
```
1 import java.io.*;
3 public class Maze {
4     private int m, n;
5     private int[][] a;
6     private static final int OPEN = 0, WALL = 1, TRIED = 2, PATH = 3;
8     public Maze(String file) {
9         // see Programming Problem 5.24 on page 197
10    }
12    public boolean isOpen(Location location) {
13        return (a[location.getY( )][location.getX( )] == OPEN);
14    }
16    public void markMoved(Location location) {
17        a[location.getY( )][location.getX( )] = PATH;
18    }
```

```
20    public void markTried(Location location) {
21        a[location.getY()][location.getX()] = TRIED;
22    }
24    public int getWidth() {
25        return n;    %return m;
26    }
28    public int getHeight() {
29        return m;    %return n;
30    }
32    public void print() {
33        char[] chars = {' ', '+', '?', 'o'};
34        for (int i = 0; i < m; i++) {
35            for (int j = 0; j < n; j++)
36                System.out.print( chars[ a[i][j] ] );
37            System.out.println();
38        }
39    }
40 }
```

5.5 연결 구현

- Stack 인터페이스에 대한 배열 구현 ArrayStack은 스택이 꽉 찼을 때 배열을 재구축해야 하므로 다소 비효율적임
- Stack 인터페이스에 대한 연결 구현: LinkedStack (리스트 5.10)

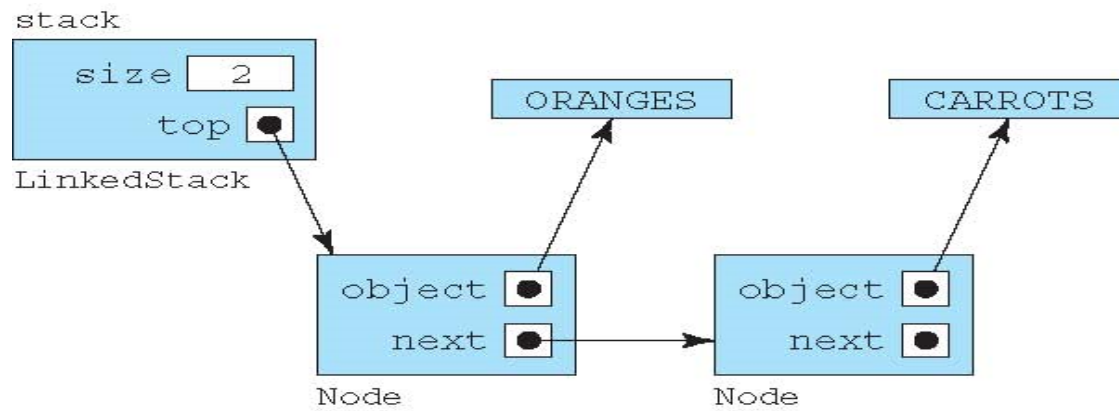
Stack 인터페이스의 연결 구현



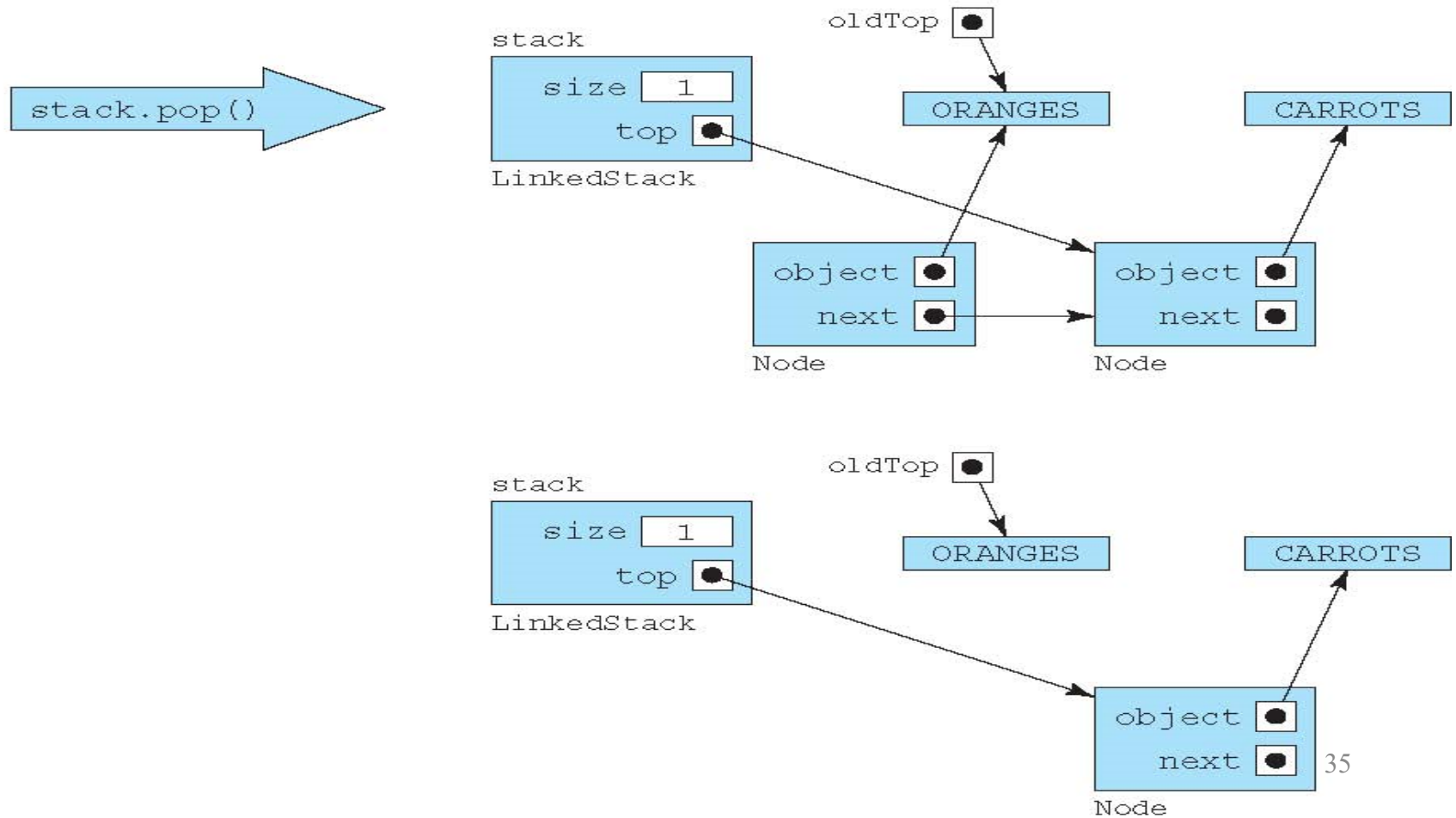
- LISTING 5.10: A LinkedStack Class

```
1 public class LinkedStack implements Stack {
2     private Node top;
3     private int size;
4
5     public boolean isEmpty() {
6         return (size == 0);
7     }
8
9     public Object peek() {
10        if (size == 0) throw new java.util.NoSuchElementException();
11        return top.object;
12    }
13
14    public Object pop() {
15        if (size == 0) throw new java.util.NoSuchElementException();
16        Object oldTop = top.object;
17        top = top.next;
18        --size;
19        return oldTop;
20    }
```

```
22  public void push(Object object) {
23      top = new Node(object,top);
24      ++size;
25  }
27  public int size() {
28      return size;
29  }
31  private static class Node {
32      Object object;
33      Node next;
34      Node(Object object, Node next) {
35          this.object = object;
36          this.next = next;
37      }
38  }
39 }
```



pop() 메소드
호출의 결과



5.6 java.util.Stack 클래스

- java.util 패키지에는 Stack 클래스가 구현되어 있음
 - java.util.Vector 클래스를 확장
- LISTING 5.11: java.util.Stack Class 일부 멤버에 대한 개요

```
1 public class Stack extends Vector {
2     public Stack()
3     public boolean empty()
4     public Object peek()
5     public Object pop()
6     public Object push(Object object)
7     public int size()
8 }
```
- 현재는 이 대신 java.util.ArrayList를 사용할 것을 권장