Tree Implementations

Chapter 24

THIRD EDITION

Data Structures and Abstractions with Java™

FRANK M. CARRANO

Contents

- The Nodes in a Binary Tree
  - An Interface for a Node
  - An Implementation of **`BinaryNode`**
- An Implementation of the ADT Binary Tree
  - Creating a Basic Binary Tree
  - The Method **`privateSetTree`**
  - Accessor and Mutator Methods
  - Computing the Height and Counting Nodes
  - Traversals

Contents

- An Implementation of an Expression Tree
  - General Trees
  - A Node for a General Tree
  - Using a Binary Tree to Represent a General Tree

Objectives

- Describe necessary operations on node within binary tree
- Implement class of nodes for binary tree
- Implement class of binary trees
- Implement an expression tree by extending class of binary trees
- Describe necessary operations on a node within general tree
- Use binary tree to represent general tree

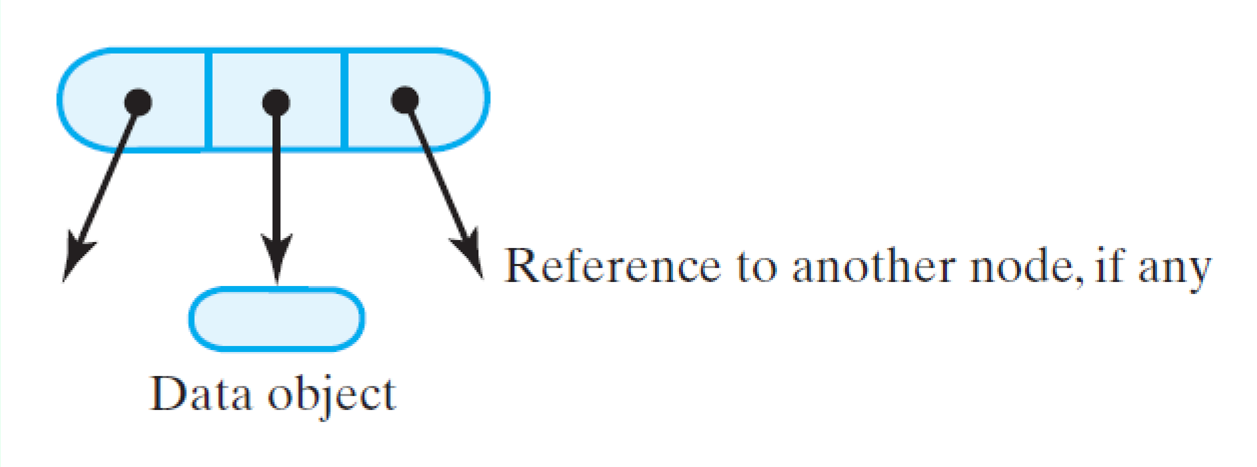


Figure 24-1 A node in a binary tree

An Interface for a Node

- Note code for node interface, Listing 24-1
- An implementation of **`BinaryNode,`** Listing 24-2
- Creating a basic binary tree
  - First draft of the class, Listing 24-3

Note: Code listing files must be in same folder as PowerPoint files for links to work

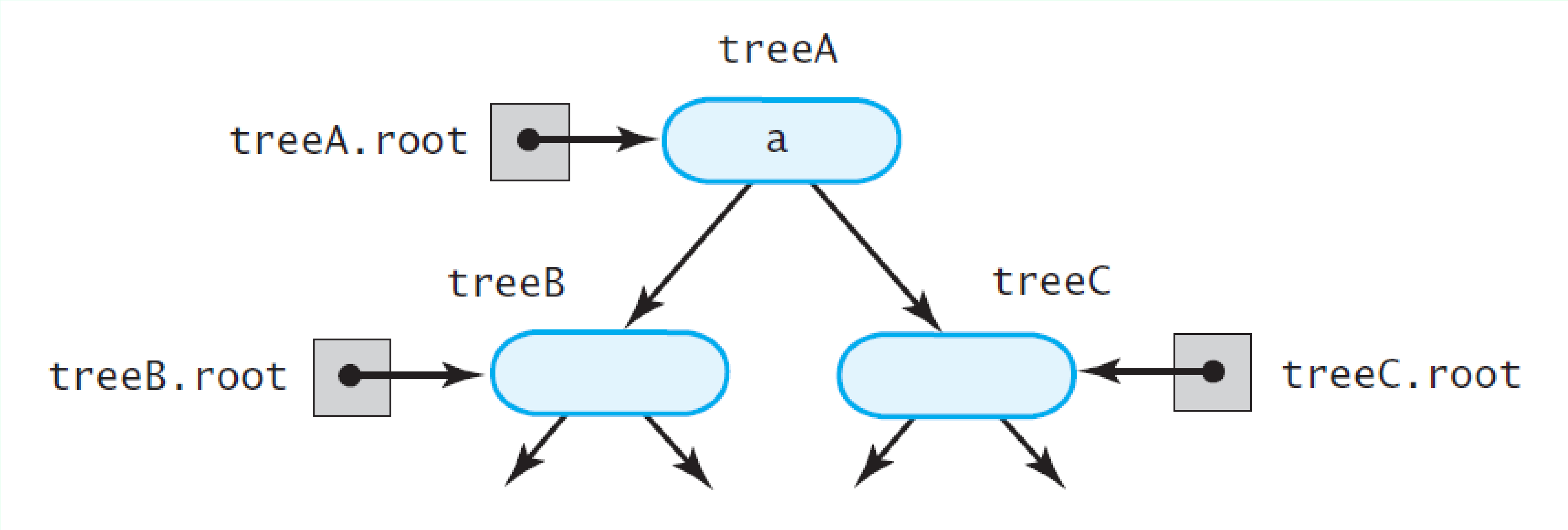


Figure 24-2 The binary tree treeA shares nodes with treeB and treeC

Q 1 In the previous method copy, are the casts to BinaryNode<T> necessary? Explain.

Q 1 In the previous method copy, are the casts to BinaryNode<T> necessary? Explain.

Yes. The fields left and right of BinaryNode (see Segment 24.3) have BinaryNode<T> as their data type, but the return type of the method copy is BinaryNodeInterface<T>.

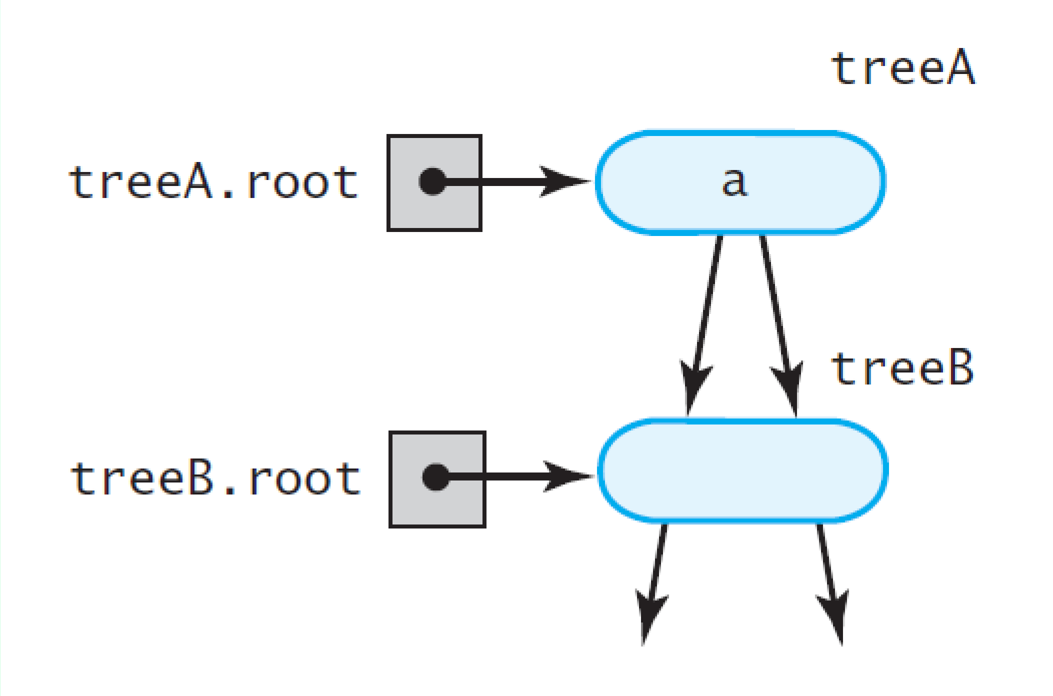


Figure 24-3 treeA has identical subtrees

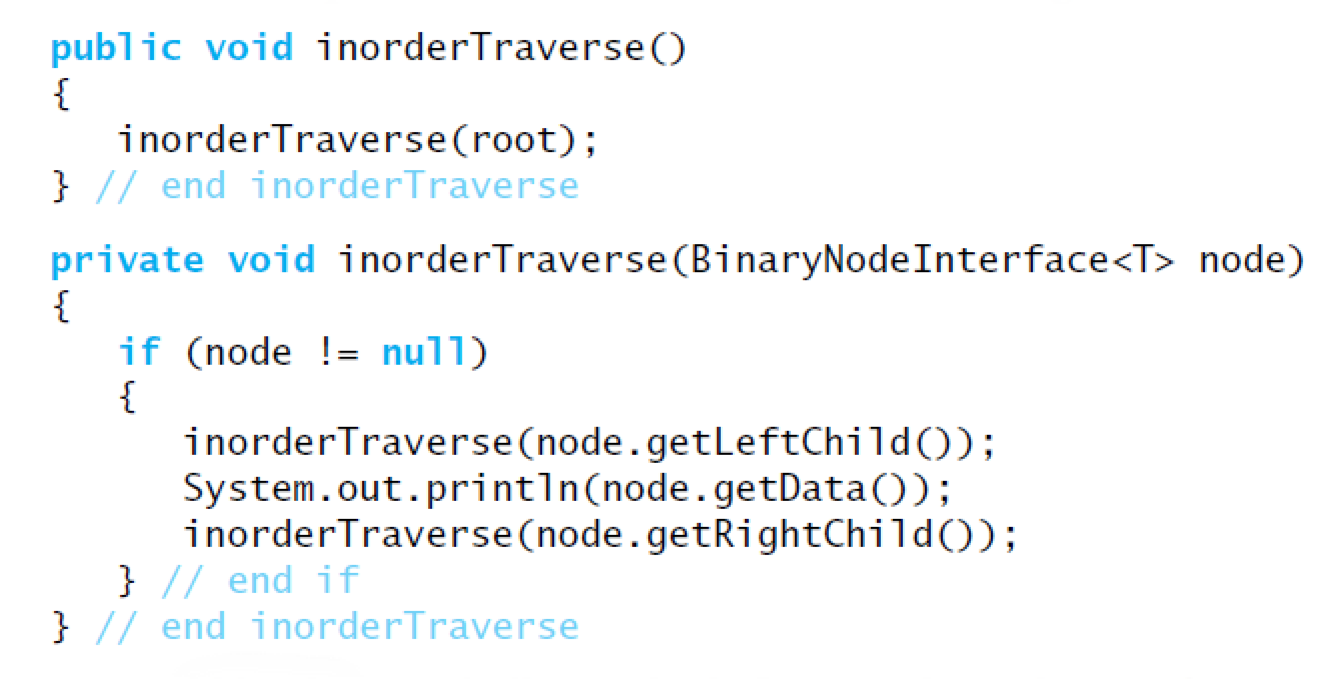
Question 2 At the end of the implementation of privateSetTree, can you set rightTree to null instead of invoking clear? Explain.

Question 2 At the end of the implementation of privateSetTree, can you set rightTree to null instead of invoking clear? Explain.

No. Setting rightTree to null affects only the local copy of the reference argument rightTree. An analogous comment applies to leftTree .

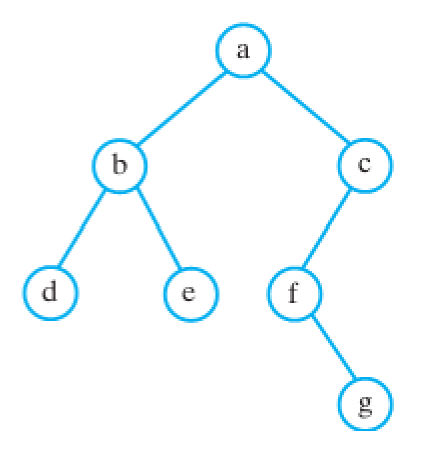
Traversing Recursively

- Inorder traversal
  - Public method for user, calls private method



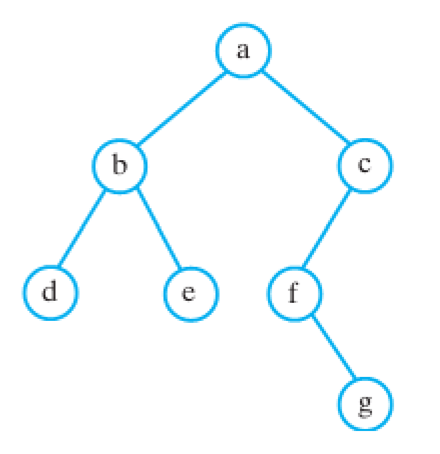
```
public void inorderTraverse()
{
   inorderTraverse(root);
} // end inorderTraverse

private void inorderTraverse(BinaryNodeInterface<T> node)
{
   if (node != null)
   {
      inorderTraverse(node.getLeftChild());
      System.out.println(node.getData());
      inorderTraverse(node.getRightChild());
   } // end if
} // end inorderTraverse
```

Question 3 Trace the method inorderTraverse with the binary tree in Figure 24-4. What data is displayed?



Question 4 Implement a recursive method preorder Traverse that displays the data in a binary tree in preorder.

Question 3 Trace the method inorderTraverse with the binary tree in Figure 24-4. What data is displayed?



The data in the objects d , b, e , a, f , g, and c is displayed on separate lines.

Question 4 Implement a recursive method preorder Traverse that displays the data in a binary tree in preorder.

```
 public void  preorderTraverse()
{             preorderTraverse(root);
}
private void preorderTraverse(BinaryNodeInterface<T> node)
{             if (node != null)
              {             System.out.println(node.getData());
                            preorderTraverse(node.getLeftChild());
                            preorderTraverse(node.getRightChild());
              }
}
```

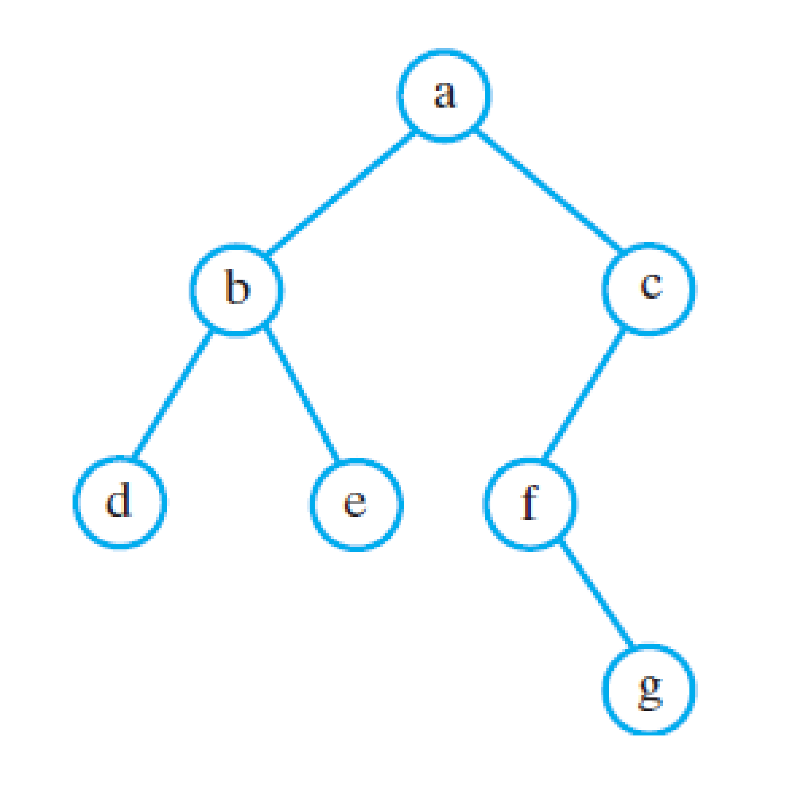


Figure 24-4 A binary tree

Traversals with An Iterator

- Iterator traversal provides more flexibility
- Class **`BinaryTree`** must implement methods in interface **`TreeIteratorInterface`**
- Possible to use a stack to do inorder traversal
  - Note example, Listing 24-A
- Private class **`InorderIterator`**, Listing 24-4

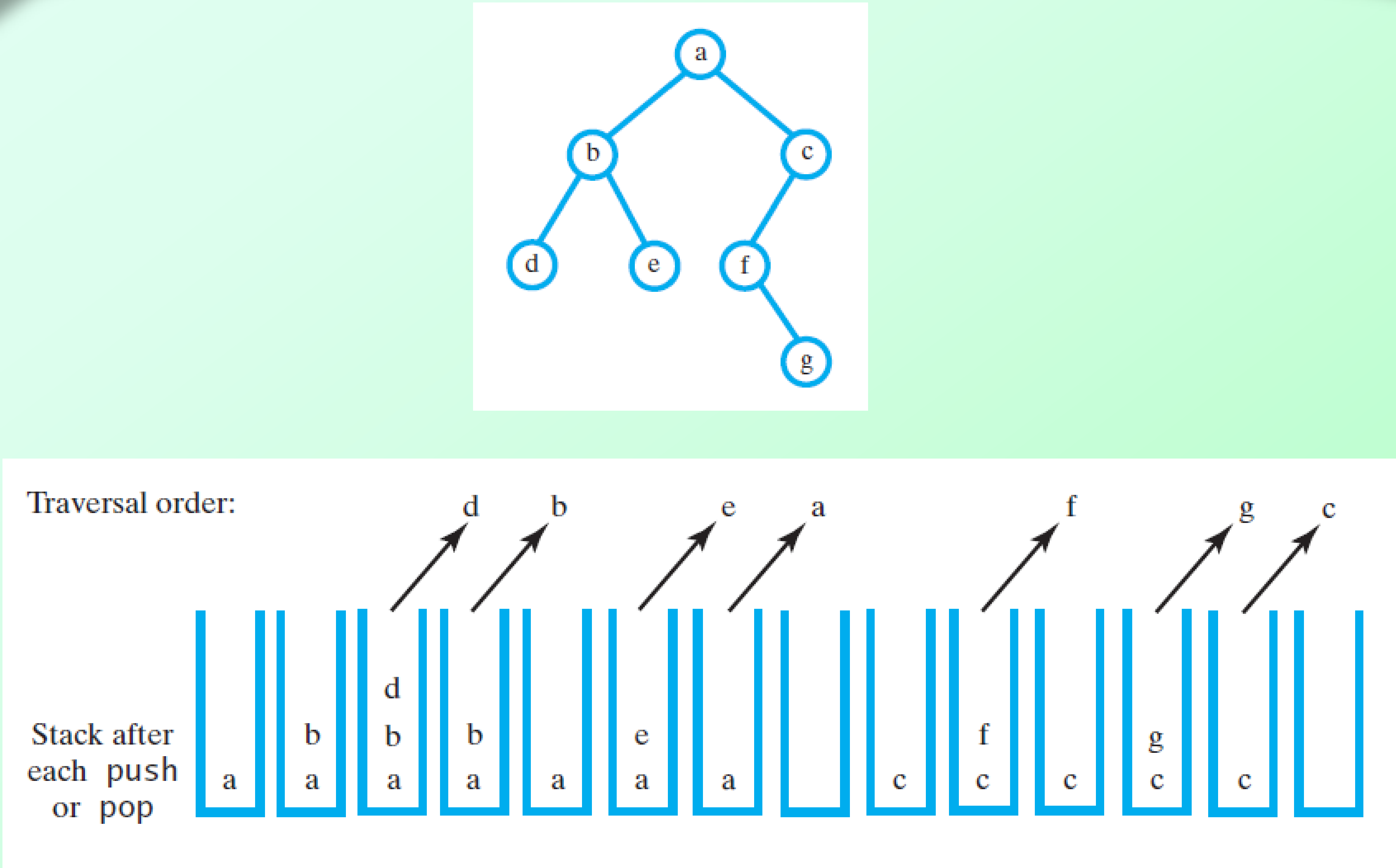


Figure 24-5 Using a stack to perform an inorder traversal of a binary tree

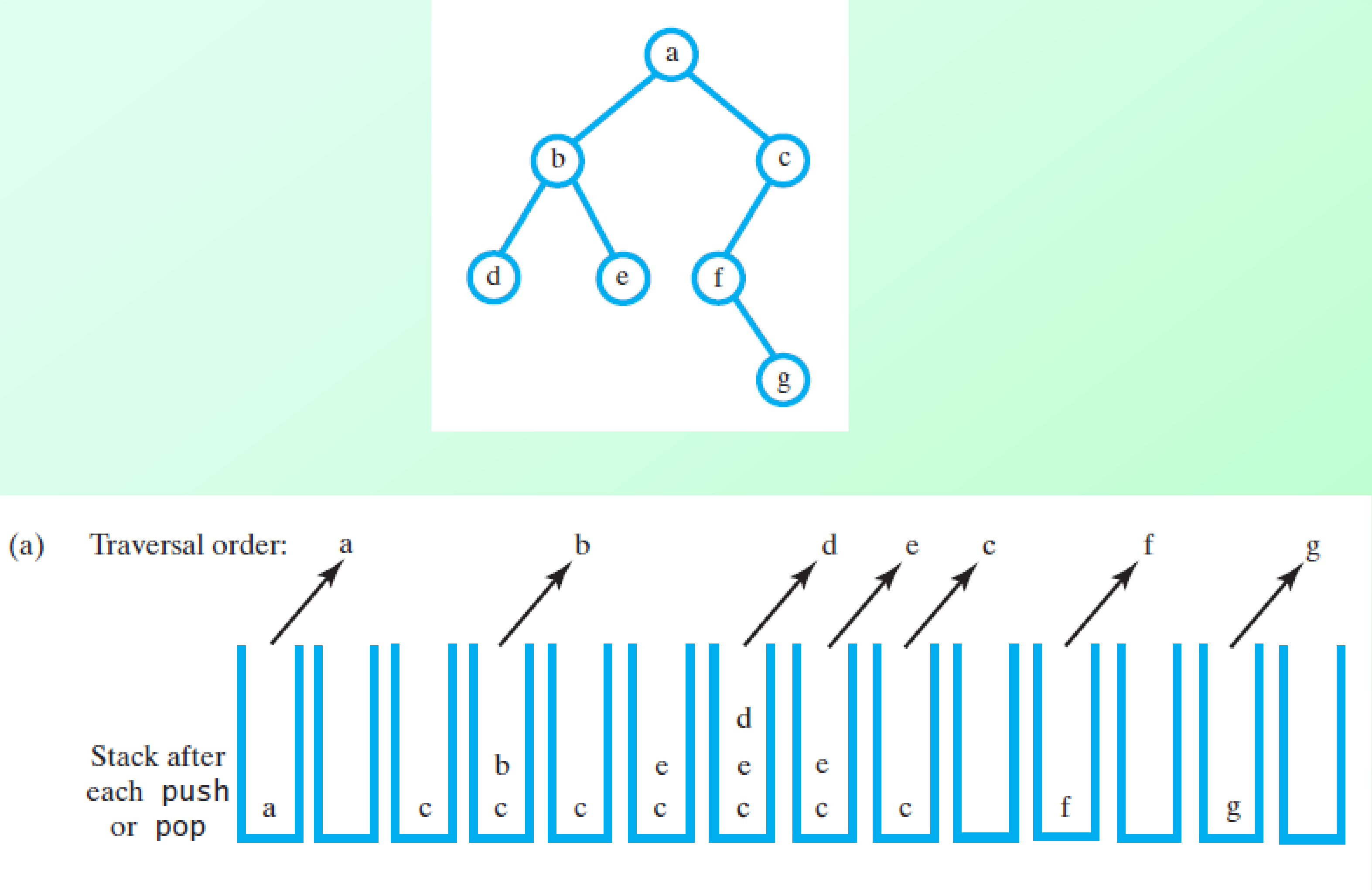

a
b
c
d
e
f
g
(a) Traversal order: a b d e c f g
Stack after each push or pop
a
c
b c
c
e c
d e c
e c
c
f
g

Figure 24-6 Using a stack to traverse a binary tree in (a) preorder;

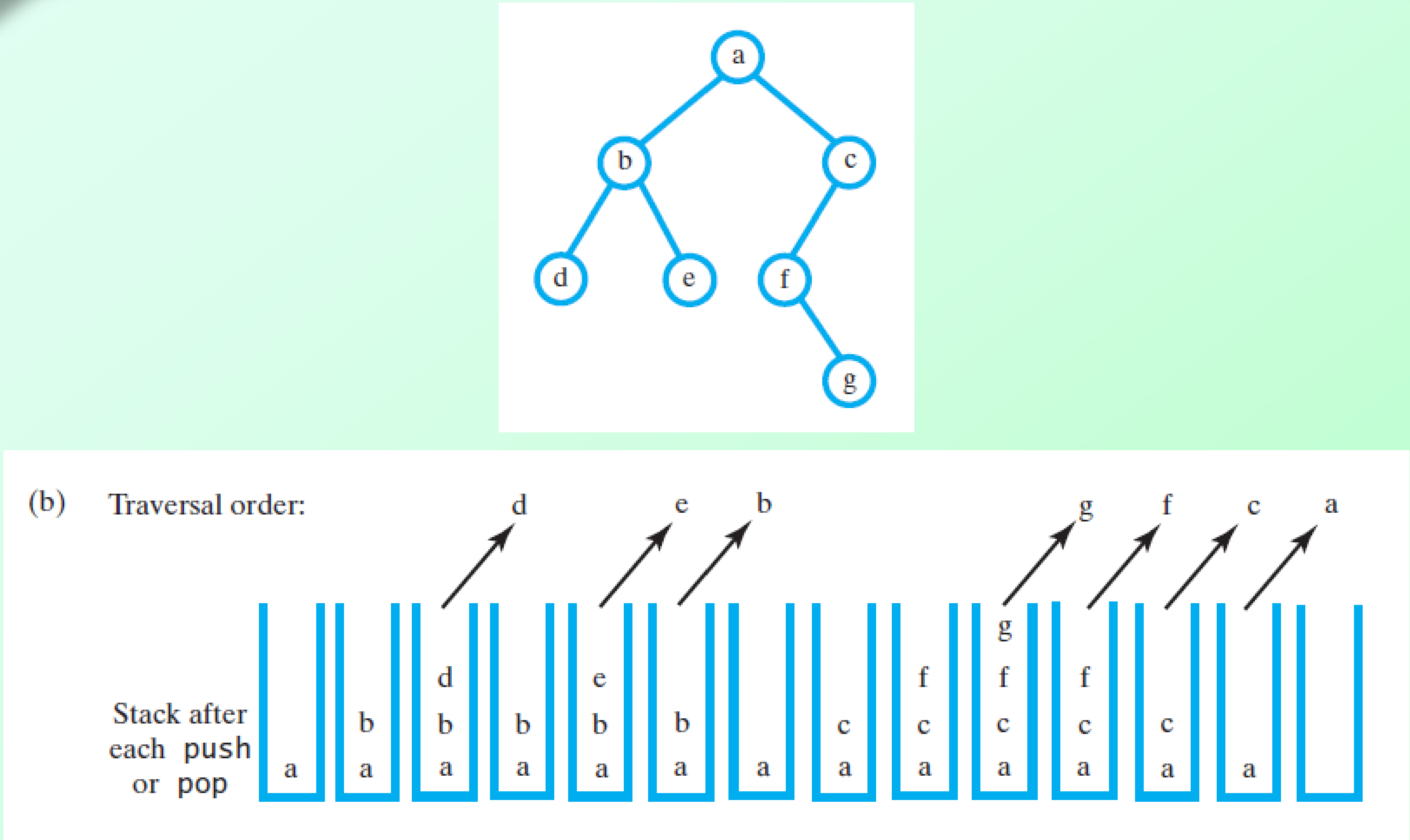
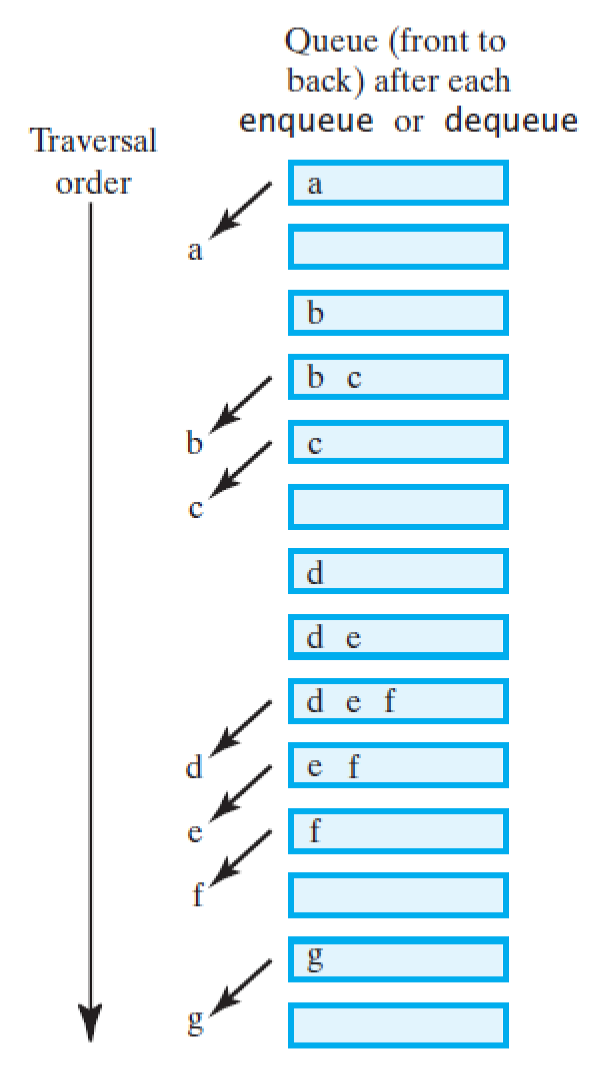


Figure 24-6 Using a stack to traverse a binary tree in (b) postorder;



Traversal order

Queue (front to back) after each enqueue or dequeue

a

b

b c

c

d

d e

d e f

e f

f

g

a

b

c

d

e

f

g

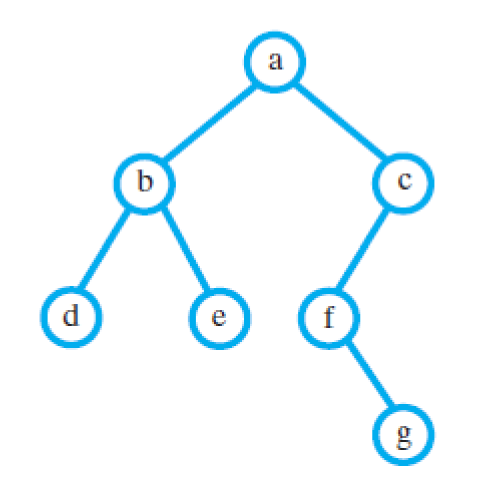
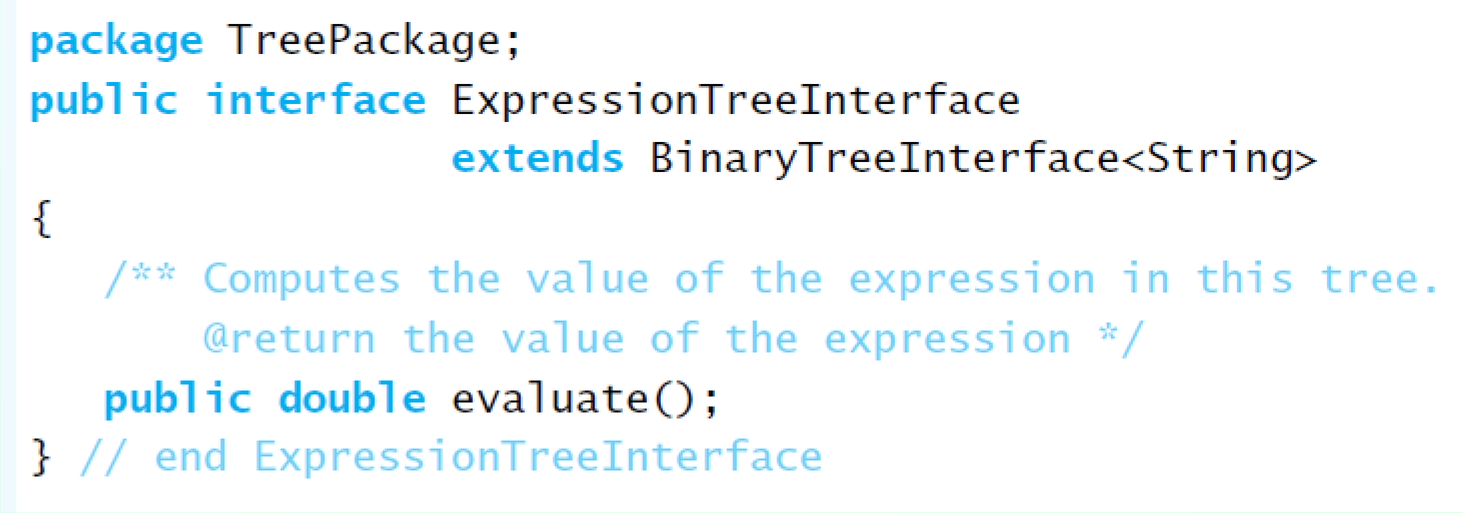


Figure 24-7 Using a queue to traverse a binary tree in level order

Implementation of an Expression Tree

- Note interface, Listing 24-5



```
package TreePackage;
public interface ExpressionTreeInterface
                extends BinaryTreeInterface<String>
{
   /** Computes the value of the expression in this tree.
       @return the value of the expression */
   public double evaluate();
} // end ExpressionTreeInterface
```

- Derive from BinaryTree, Listing 24-6

Question 6 Trace the method evaluate for the expression tree in Figure 23-14c of the previous chapter. What value is returned? Assume that a is 3, b is 4, and c is 5.

Question 6 Trace the method evaluate for the expression tree in Figure 23-14c of the previous chapter. What value is returned? Assume that a is 3, b is 4, and c is 5.

27

Node for a General Tree

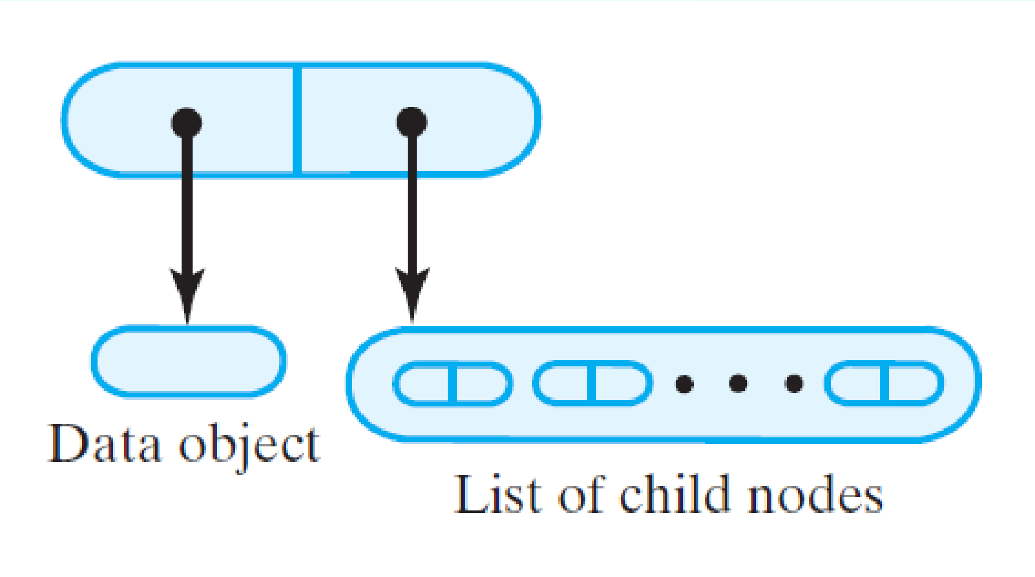
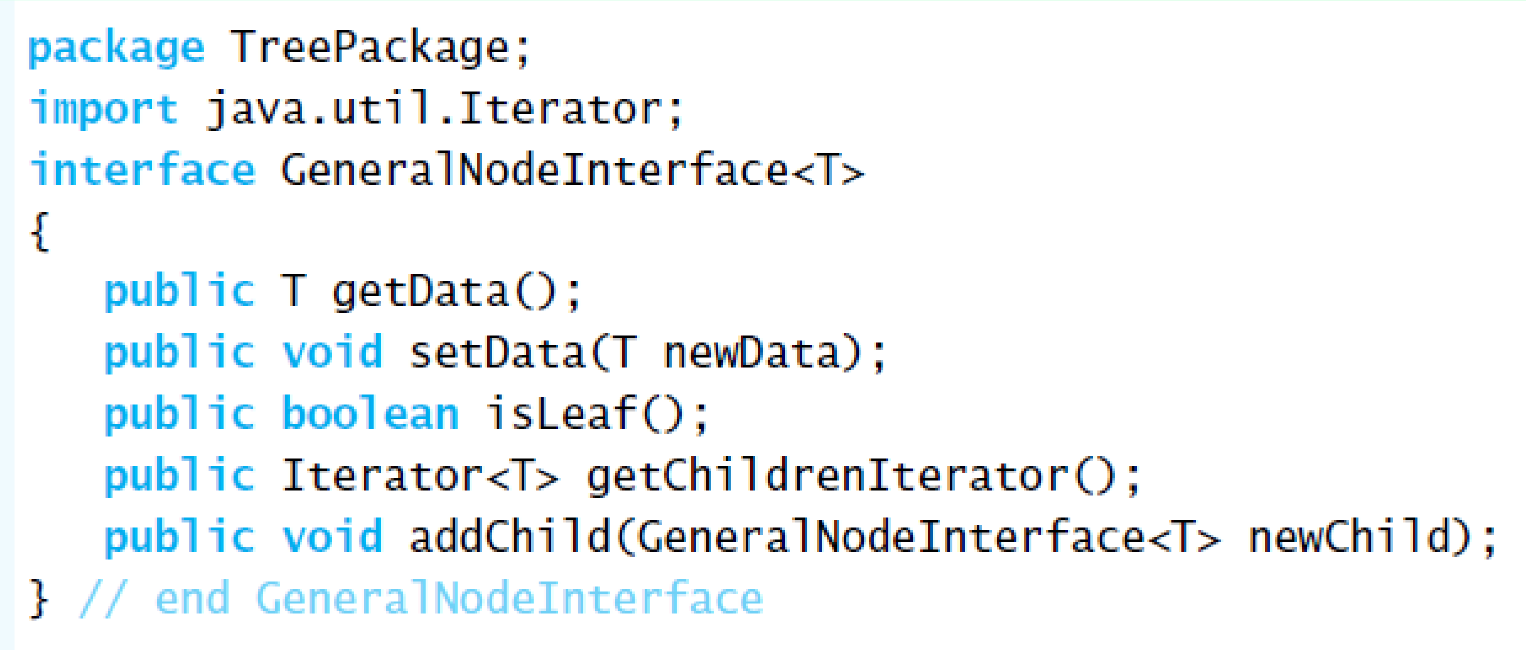


Figure 24-8 A node for a general tree

Node for a General Tree

- Interface, Listing 24-7



```
package TreePackage;
import java.util.Iterator;
interface GeneralNodeInterface<T>
{
   public T getData();
   public void setData(T newData);
   public boolean isLeaf();
   public Iterator<T> getChildrenIterator();
   public void addChild(GeneralNodeInterface<T> newChild);
} // end GeneralNodeInterface
```

Using a Binary Tree to Represent a General Tree

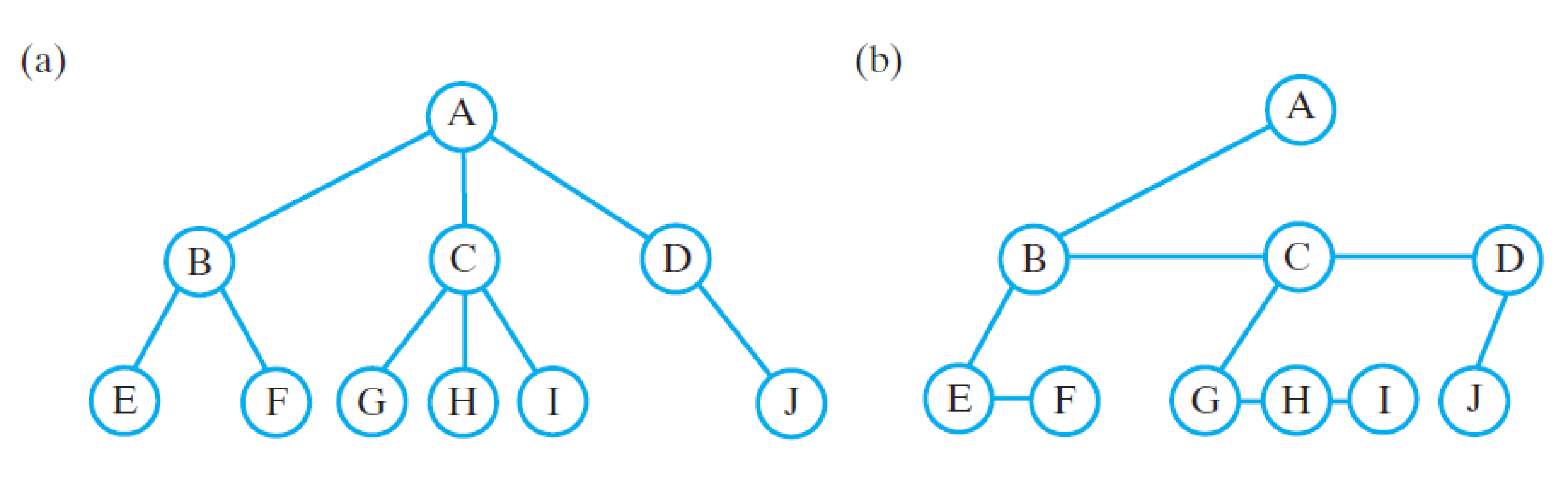


Figure 24-9 (a) A general tree; (b) an equivalent binary tree;

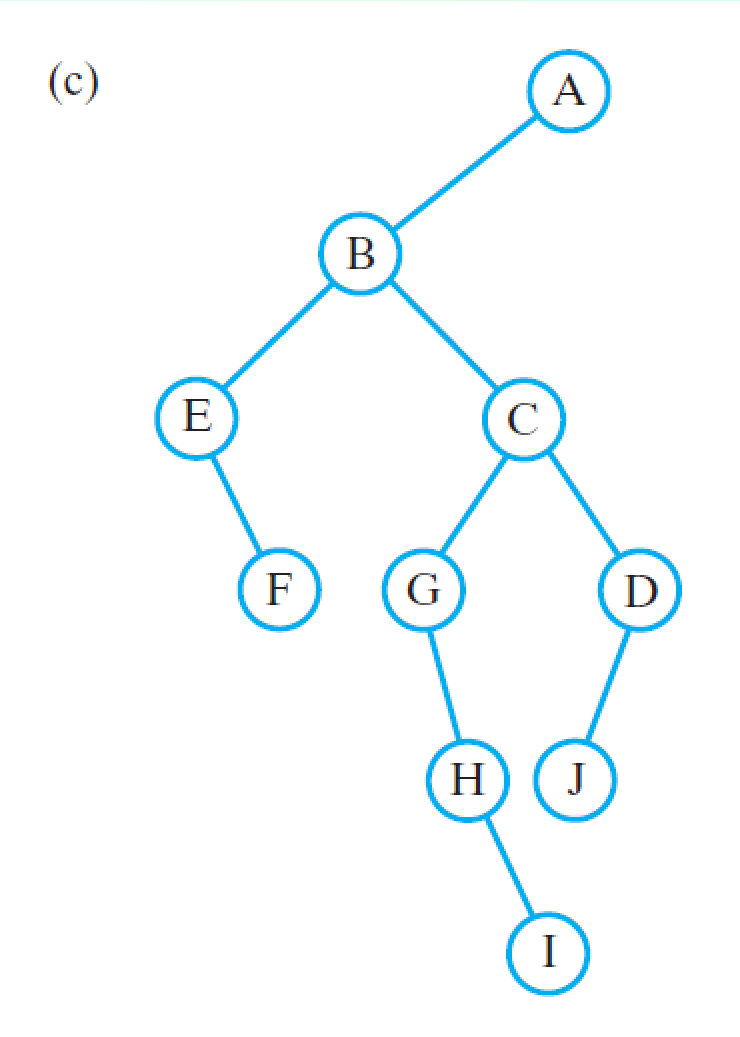
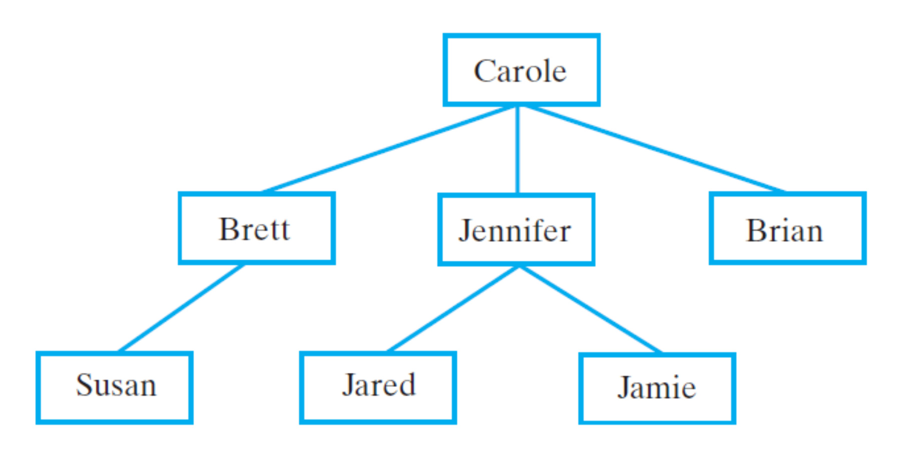
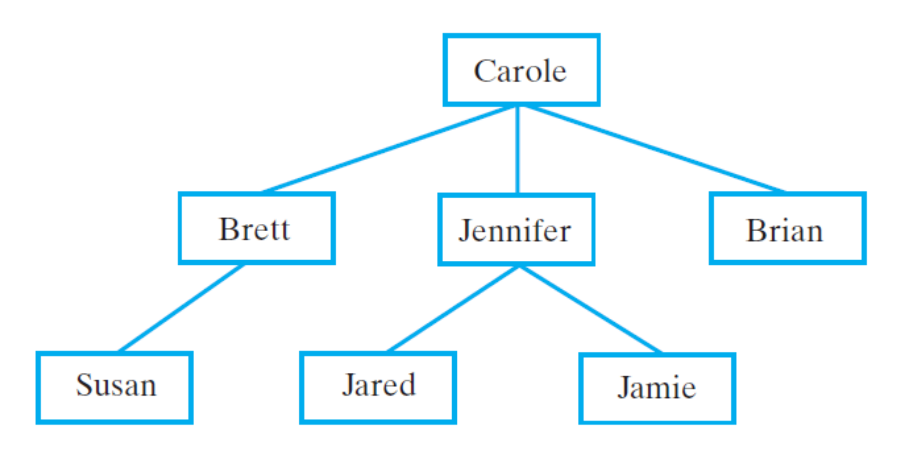


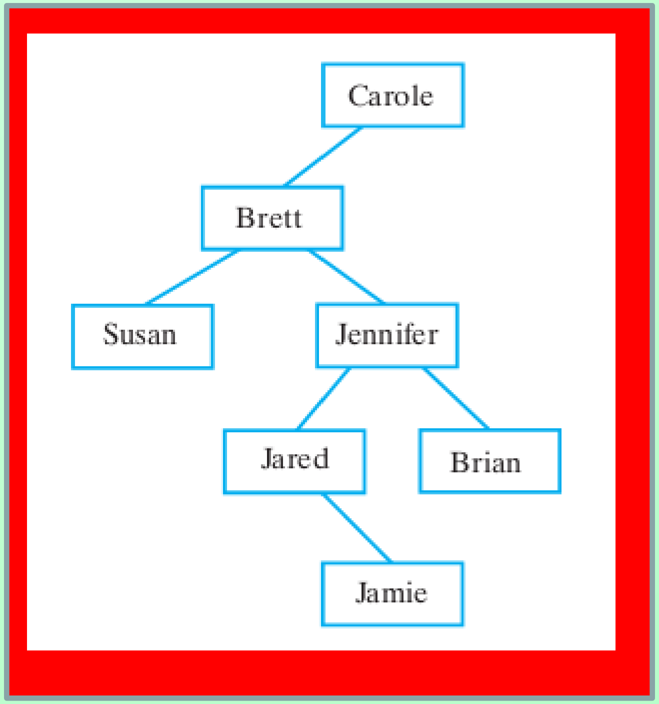
Figure 24-9 (c) a more conventional view of the same binary tree

Q 7 What binary tree can represent the general tree in Fig. 23-1 of the previous chapter?



Q 7 What binary tree can represent the general tree in Fig. 23-1 of the previous chapter?





End

Chapter 24

THIRD EDITION

Data Structures and Abstractions with Java™

FRANK M. CARRANO