問 11　次の Java プログラムの説明及びプログラムを読んで，設問1～3に答えよ。
（Java プログラムで使用する API の説明は，この冊子の末尾を参照してください。）

〔プログラムの説明〕

ディレクトリパス（以下，パスという）から木構造を生成するプログラムである。

パスは，木構造をもつファイルシステムにおいてディレクトリを特定するために利用される文字列であり，ディレクトリの名前を“/”で区切って並べて表す。“/”で始まるパスを絶対パスという。絶対パスはルートディレクトリを起点として表したパスである。“/”で始まらないパスを相対パスという。相対パスは任意のノードを起点として表したパスである。

このプログラムが生成する木構造中の各ノードは，それぞれが一つのディレクトリを表し，ルートノードはルートディレクトリを表す。

図1に木構造の例を示す。図1中の楕円一つはノード一つに対応し，“ と ”で囲まれた文字列はノードの名前を表す。ルートノードの名前は空文字列とする。例えば，ノード usr を特定する絶対パスは“/usr”であり，ノード usr を起点とする相対パス“local/lib”が特定するノードは，絶対パス“/usr/local/lib”が特定するノードと同じノード lib である。

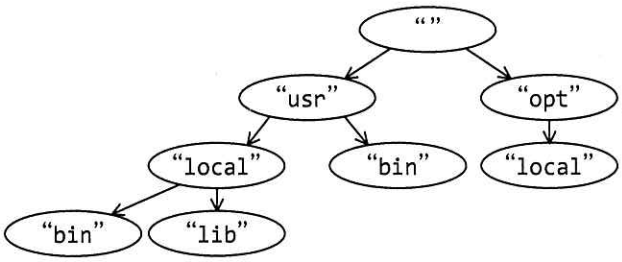

図 1　木構造の例

クラス DirectoryNode は木構造を構成するノードを表すクラスであり，一つのインスタンスが一つのノードを表す。フィールド name はノードの名前を，フィールド parent は親ノードへの参照を，フィールド children は子ノードのリストを保持する。引数を取らないコンストラクタはルートノードを生成する。

クラス DirectoryNode は次のメソッドをもつ。

(1) public DirectoryNode add(String path)

引数 path で与えられたパスが，このノードを起点に一つのノードを特定できるように木構造を拡張し，パスが特定するノードを返す。引数 path が空文字列又は絶対パスを表すなら，IllegalArgumentException を投げる。

パス中の連続する“/”は一つの“/”として扱い，末尾の“/”は無視する。つまり，パス“local//lib/”は“local/lib”とみなす。

(2) public String path()

このノードを特定する絶対パスを表す文字列（末尾は常に“/”）を返す。

例えば，図 1 の最下段のノード lib でこのメソッドを呼ぶと，“/usr/local/lib/”を返す。

(3) public List<DirectoryNode> find(String name)

このノードが保持する各子ノードを頂点とする全ての部分木から，引数 name で与えられた名前をもつノードを全て探し，見つかったノードをリストで返す。

クラス DirectoryNodeTester はテスト用のプログラムである。実行結果を図 2 に示す。

```
usr
opt
/usr/local/
/opt/local/
IllegalArgument
```

図 2　クラス DirectoryNodeTester の実行結果

〔プログラム 1〕

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class DirectoryNode {
   public final DirectoryNode parent;
   public final String name;
   public final List<DirectoryNode> children =
      new ArrayList<DirectoryNode>();

   public DirectoryNode() {
      this( a );
   }
```

```
private DirectoryNode(String name, DirectoryNode parent) {
    this.name = name;
    this.parent = parent;
}

public DirectoryNode add(String path) {
    if ( [ b ] ) {
        throw new IllegalArgumentException();
    }
    // 第1引数は文字列pathに含まれるディレクトリのリストである
    return add(Arrays.asList(path.split("/+")), 0);
}

private DirectoryNode add(List<String> path, int i) {
    DirectoryNode child = findChild(path.get(i));
    if (child == null) {
        children.add(child = new DirectoryNode(path.get(i), this));
    }
    if (path.size() > i + 1) {
        return child.add(path, i + 1);
    }
    return child;
}

public String path() {
    if ( [ c ] ) { // 自ノードがルートノードなら"/"を返す
        return "/";
    }
    return parent.path() + name + "/";
}

private DirectoryNode findChild(String name) {
    for (DirectoryNode child : children) {
        if (child.name.equals(name)) {
            return child;
        }
    }
    return null;
}

public List<DirectoryNode> find(String name) {
    List<DirectoryNode> ret = new ArrayList<DirectoryNode>();
    DirectoryNode node = findChild(name);
    if (node != null) {
        ret.add( [ d ] );
    }
    for (DirectoryNode child : children) {
        ret.addAll(child.find( [ e ] ));
    }
    return ret;
```

```
    }
}
```

〔プログラム2〕

```
public class DirectoryNodeTester {
    public static void main(String[] args) {
        try {
            DirectoryNode root = new DirectoryNode();
            DirectoryNode usr = root.add("usr");
            usr.add("bin");
            usr.add("local/bin");
            usr.add("local/lib");
            root.add("opt/local");
            for (DirectoryNode n : root.children) {
                System.out.println(n.name);
            }
            for (DirectoryNode n : root.find("local")) {
                System.out.println(n.path());
            }
            usr.add("");
        } catch (IllegalArgumentException e) {
            System.out.println("IllegalArgument");
        }
    }
}
```


設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

aに関する解答群

ア　""　　イ　"", null　　ウ　null　　エ　null, ""

bに関する解答群

ア　!path.startsWith("/") && !"".equals(path)

イ　!path.startsWith("/") || !"".equals(path)

ウ　path.startsWith("/") && "".equals(path)

エ　path.startsWith("/") || "".equals(path)

cに関する解答群

ア　parent != ""　　イ　parent != null

ウ　parent == ""　　エ　parent == null

d, eに関する解答群

ア child	イ name	ウ node
エ null	オ parent	カ root

設問2 クラス DirectoryNodeTester を実行したときに生成される，クラス DirectoryNode のインスタンスの個数として正しい答えを，解答群から選べ。

解答群

ア 6　　イ 7　　ウ 8　　エ 9　　オ 10

設問3 次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

クラス DirectoryNode のメソッド path が返す文字列の末尾は常に“/”である。これを，このメソッドが呼ばれたインスタンスがルートノードであるか，子ノードをもつときにだけ末尾が“/”になるように，メソッド path の最後の return 文の直前に次の3行を挿入した。クラス DirectoryNodeTester の実行結果は図3となる。

```
if ( [ f ] ) {
   return parent.path() + name;
}
```

```
usr
opt
/usr/local/
/opt/local
IllegalArgument
```

図3 クラス DirectoryNode 変更後のクラス DirectoryNodeTester の実行結果

fに関する解答群

ア !children.isEmpty()	イ children != null
ウ children == null	エ children.isEmpty()