processAnOperator 方法（第 90 ～ 103 行）用来处理一个运算符。该方法从 operatorStack 中弹出一个运算符（第 92 行）并且从 operandStack 中弹出两个操作数（第 93 ～ 94 行）。依据所弹出的运算符，该方法完成对应的操作，然后将操作结果压回 operandStack 中（第 96、98、100 和 102 行）。

### 复习题

20.23 EvaluateExpression 程序可以对表达式 "1 + 2"、"1 + 2"、"(1) + 2"、"((1)) + 2" 以及 "(1 + 2)" 求值吗？

20.24 使用 EvaluateExpression 程序对 "3 + (4 + 5)*(3 + 5) + 4 * 5" 求值时，给出栈中内容的变化。

20.25 如果输入表达式 "4 + 5 5 5"，程序将显示 10。如果修改这个问题？

## 关键术语

collection（合集）
comparator（比较器）
convenience abstract class（便利抽象类）
data structure（数据结构）
linked list（链表）
list（线性表）
priority queue（优先队列）
queue（队列）

## 本章小结

1. Java 合集框架支持集合、线性表、队列和映射表，它们分别定义在接口 Set、List、Queue 和 Map 中。
2. 线性表用于存储一个有序的元素合集。
3. 除去 PriorityQueue，Java 合集框架中的所有实例类都实现了 Cloneable 和 Serializable 接口。所以，它们的实例都是可克隆和可序列化的。
4. 若要在合集中存储重复的元素，就需要使用线性表。线性表不仅可以存储重复的元素，而且允许用户指定存储的位置。用户可以通过下标来访问线性表中的元素。
5. Java 合集框架支持两种类型的线性表：数组线性表 ArrayList 和链表 LinkedList。ArrayList 是实现 List 接口的可变大小的数组。ArrayList 中的所有方法都是在 List 接口中定义的。LinkedList 是实现 List 接口的一个链表。除了实现了 List 接口，该类还提供了可从线性表两端提取、插入以及删除元素的方法。
6. Comparator 可以用于比较没有实现 Comparable 接口的类的对象。
7. Vector 类继承了 AbstractList 类。从 Java 2 开始，Vector 类和 ArrayList 是一样的，所不同的是它所包含的访问和修改向量的方法是同步的。Stack 类继承了 Vector 类，并且提供了几种对栈进行操作的方法。
8. Queue 接口表示队列。PriorityQueue 类为优先队列实现 Queue 接口。

## 测试题

回答位于网址 www.cs.armstrong.edu/liang/intro10e/quiz.html 的本章测试题。

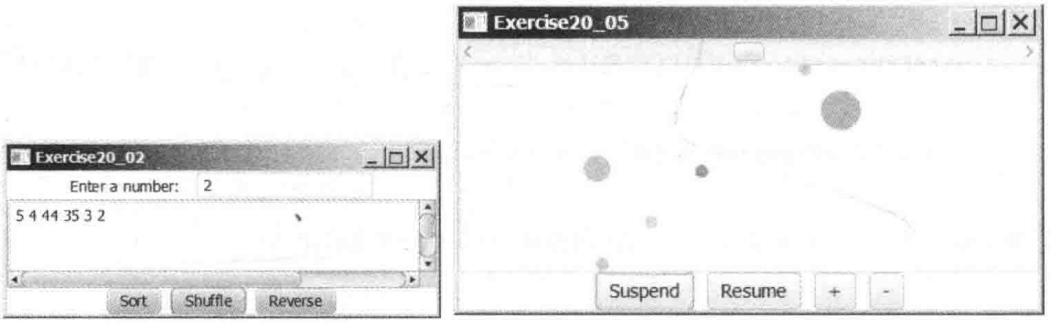
## 编程练习题

20.2 ～ 20.7 节

*20.1 （按字母序的升序显示单词）编写一个程序，从文本文件读取单词，并按字母的升序显示所有的

单词（可以重复）。单词必须以字母开始。文本文件作为命令行参数传递。

*20.2 （对链表中的数字进行排序）编写一个程序，让用户从图形用户界面输入数字，然后在文本区域显示它们，如图 20-17a 所示。使用链表存储这些数字，但不要存储重复的数值。添加按钮 Sort、Shuffle 和 Reverse，分别对这个线性表进行排序、打乱顺序与颠倒顺序操作。




a) b)

图 20-17 a）数字保存在线性表中并显示在一个文本区域内；b）相撞的球结合在一起

*20.3 （猜首府）改写编程练习题 8.37，保存州和首府的匹配对，以随机显示问题。

*20.4 （对面板上的点进行排序）编写一个程序，满足下面的要求：

- 定义一个名为 Point 的类，它的两个数据域 x 和 y 分别表示点的 $x$ 坐标和 $y$ 坐标。实现 Comparable 接口用于比较点的 $x$ 坐标。如果两个点的 $x$ 坐标一样，则比较它们的 $y$ 坐标。
- 定义一个名为 CompareY 的类实现 Comparator<Point>。实现 compare 方法来通过 $y$ 坐标值比较两个点。如果 $y$ 坐标值一样，则比较它们的 $x$ 坐标值。
- 随机创建 100 个点，然后使用 Arrays.sort 方法分别以它们 $x$ 坐标的升序和 $y$ 坐标的升序显示这些点。

***20.5 （合并碰撞的弹球）20.7 节的示例中显示了多个弹球。扩充该例子来进行碰撞检测。一旦两个球相撞，移除后面加入面板的那个球，并且将它的半径加到另外一个球上，如图 20-17b 所示。使用 Suspend 按钮来暂停动画，以及 Resume 按钮来继续动画。添加一个鼠标按下处理器，从而在鼠标按在球上的时候移除这个球。

20.6 （在链表上使用遍历器）编写一个测试程序，在一个链表上存储 500 万个整数，测试分别使用 interator 和使用 get(index) 方法的遍历时间。

***20.7 （游戏：猜字游戏）编程练习题 7.35 给出了流行的猜字游戏的控制台版本。编写一个 GUI 程序让用户来玩这个游戏。用户通过一次输入一个字母来猜单词，如图 20-18 所示。如果用户 7 次都没猜对，被吊的人就摆动起来。一旦完成一个单词，用户就可以按 Enter 键继续猜另一个单词。

**20.8 （游戏：彩票）修改编程练习题 3.15，如果用户输入的两个数字在彩票号码之中，增加额外的 2000 美元。（提示：对彩票中的三个数字和用户输入的三个数字进行排序，并分别存入两个线性表，然后使用 Collection 的 containsAll 方法来检测用户输入的两个数字是否在彩票数字中。）

20.8 ～ 20.10 节

***20.9 （首先移除最大的球）修改程序清单 20-6，使得一个球在被创建的时候赋给一个 2 ～ 20 的随机半径。当单击“-”按钮时，最大的一个球被移除。

20.10 （在优先队列上进行集合操作）创建两个优先队列，{"George", "Jim", "John", "Blake", "Kevin", "Michael"} 和 {"George", "Katie", "Kevin", "Michelle", "Ryan"}，求它们的并集、差集和交集。

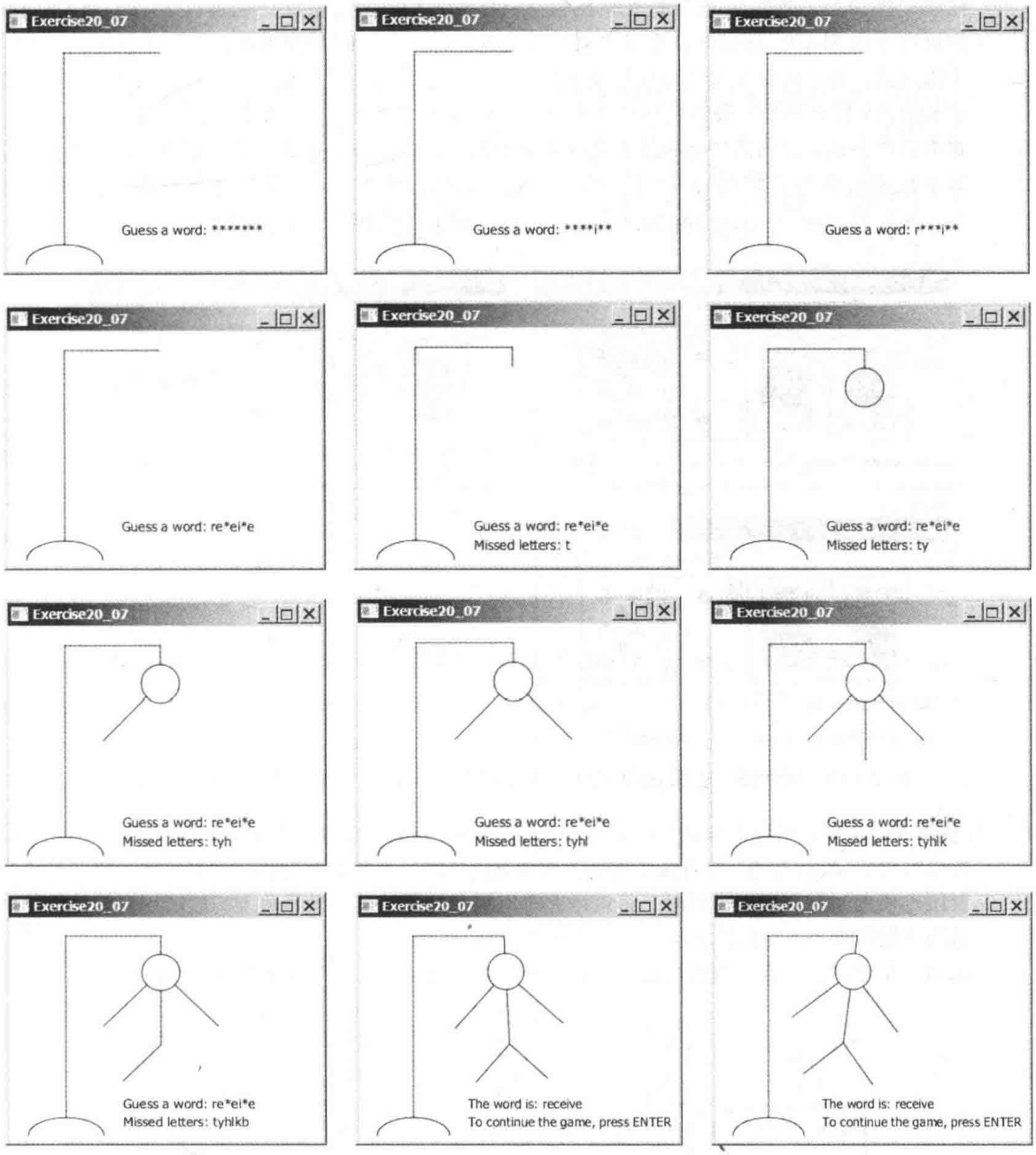



图 20-18 程序显示一个猜字游戏

*20.11 （编组符号匹配）Java 程序包含各种编组符号对，例如：

- 圆括号：( 和 )
- 花括号：{ 和 }
- 方括号：[ 和 ]

请注意编组符号不能交错。例如，(a{b)} 是不合法的。编写一个程序来检测一个 Java 源程序中是否编组符号都是正确匹配的。将源代码文件名字作为命令行参数传递。

20.12 （克隆 PriorityQueue）定义 MyPriorityQueue 类，继承自 PriorityQueue 并实现 Cloneable 接口和实现 clone() 方法来克隆一个优先队列。

**20.13 （游戏：24 点扑克牌游戏）24 点游戏是指从 52 张牌中任意选取 4 张扑克牌，如图 20-19 所示。

注意，将两个王排除在外。每张牌表示一个数字。A、K、Q 和 J 分别表示 1、13、12 和 11。你可以单击 Shuffle 按钮来获取 4 张新的扑克牌。输入这 4 张扑克牌牌面的 4 个数字构成的一个表达式。每个数字必须使用且只能使用一次。可以在表达式中使用运算符（加法、减法、乘法和除法）以及括号。表达式必须计算出 24。在输入表达式之后，单击 Verify 按钮来检查表达式中的数字是否是当前所选择的扑克牌牌面上的数，并检查表达式的结果是否正确。检查结果显示在 Shuffle 按钮前面的一个标签中。假设图像以黑桃、红心、方块和梅花的顺序存储在名为 1.png，2.png，…，52.png 的文件中，这样，前 13 个图像就是黑桃的 1，2，3，…，13。

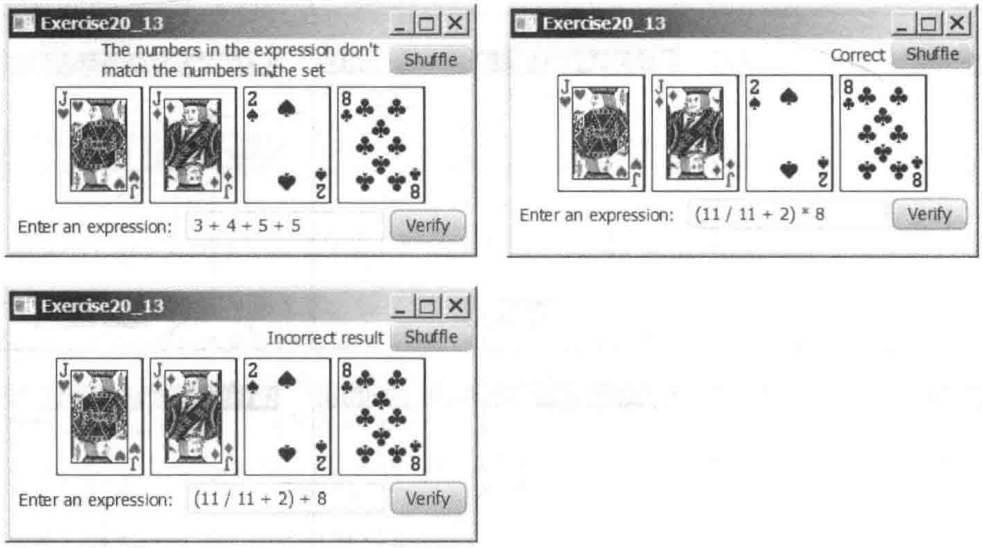
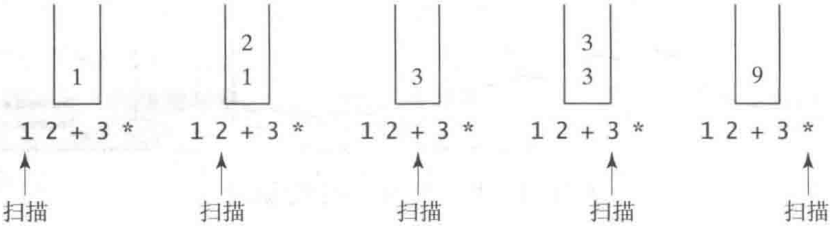



图 20-19 用户输入由牌面数字组成的表达式，并单击 Verify 按钮来检查结果

**20.14 （*后缀表示法*）后缀表示法是一种不使用括号编写表达式的方法。例如，表达式 (1 + 2) * 3 可以写为 1 2 + 3 *。后缀表达式是使用栈来计算的。从左到右扫描后缀表达式，将变量或常量压入栈内，当遇到运算符时，将该运算符应用在栈顶的两个操作数上，然后用运算结果替换这两个操作数。下面的图演示了如何计算 1 2 + 3 *。

编写一个程序，计算后缀表达式，将后缀表达式作为一个字符串的命令行参数传递。




***20.15 （*游戏：24 点扑克牌游戏*）改进编程练习题 20.13，如果表达式存在，那就让计算机显示它，如图 20-20 所示；否则，报告这样的表达式不存在。将显示验证结果的标签置于 UI 的底部。表达式必须使用所有 4 张扑克牌并且值等于 24。

**20.16 （*将中缀转换为后缀*）使用下面的方法头编写方法，将中缀表达式转换为一个后缀表达式：

```
public static String infixToPostfix(String expression)
```

例如，该方法可以将中缀表达式 (1+2)*3 转换为 1 2 + 3 *，将 2*(1+3) 转换为 2 1 3 + *。

***20.17 （*游戏：24 点扑克牌游戏*）此练习题是编程练习题 20.13 中描述的 24 点扑克牌游戏的变体。编写一个程序，检查是否有这 4 个给定数的 24 点的解决方案。该程序让用户输入 1 ～ 13 的 4 个

值，如图 20-21 所示。然后用户可以单击 Solve 按钮来显示解决方案，若不存在解决方案，就提示“不存在解决方案”。

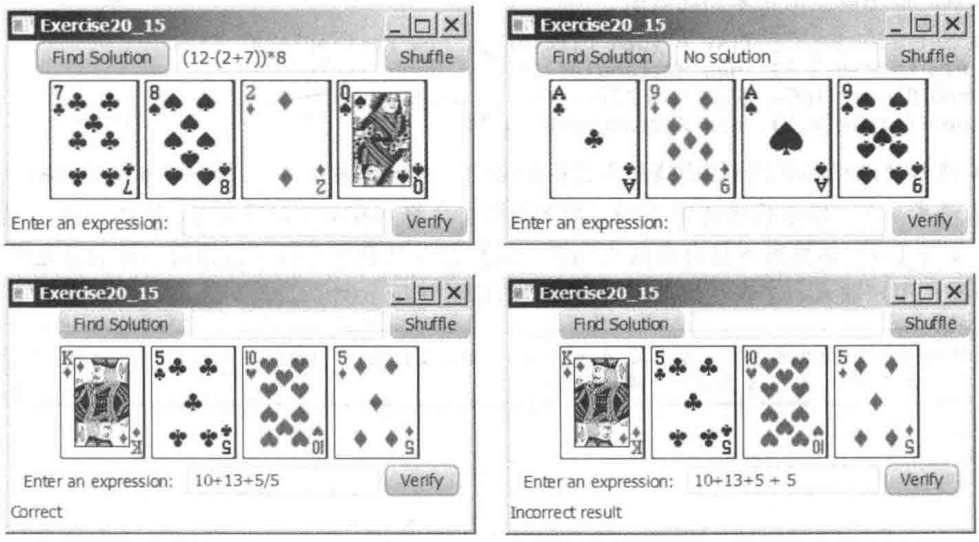



图 20-20 如果存在一个解决方案，程序可以自动找到它

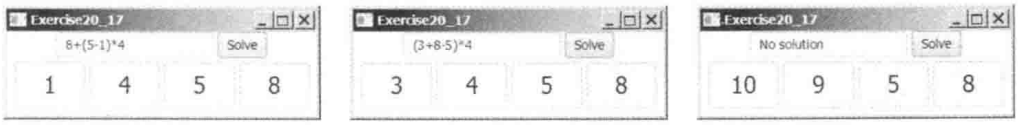



图 20-21 用户输入 4 个数字，然后程序找出解决方案

*20.18 （目录大小）程序清单 20-7 使用递归方法来找到一个目录大小。重写该方法，不使用递归。程序应该使用一个队列来存储一个目录下的所有子目录。算法可以如下描述：

```
long getSize(File directory) {
  long size = 0;
  add directory to the queue;

  while (queue is not empty) {
    Remove an item from the queue into t;
    if (t is a file)
      size += t.length();
    else
      add all the files and subdirectories under t into the
        queue;
  }

  return size;
}
```

***20.19 （游戏：24 点游戏有解的比例）回顾编程练习题 20.13 介绍的 24 点游戏，从 52 张牌中选择 4 张牌，这 4 张牌可能没有能得到 24 点的解决方案。从 52 张牌中选择 4 张牌的所有可能的挑选次数是多少？在这些所有可能的挑选中，有多少可以得到 24 点？成功的几率（即（可得到 24 点的挑选次数）/（所有可能的挑选次数））是多少？编写一个程序，找出这些答案。

*20.20 （目录大小）重写编程练习题 18.28，使用栈而不是使用队列来解决这个问题。

*20.21 （使用 Comparator）使用选择排序和比较器，编写以下通用的方法。

```
public static <E> void selectionSort(E[] list,
  Comparator<? super E> comparator)
```

编写一个测试程序，创建一个具有 10 个 GeometricObject 对象的数组，并且使用程序清单 20-4 介绍的 GeometricObjectComparator 调用该方法对元素进行排序。显示排好序的元素。使用以下语句来创建数组。

```
GeometricObject[] list = {new Circle(5), new Rectangle(4, 5),
  new Circle(5.5), new Rectangle(2.4, 5), new Circle(0.5),
  new Rectangle(4, 65), new Circle(4.5), new Rectangle(4.4, 1),
  new Circle(6.5), new Rectangle(4, 5)};
```

*20.22 （非递归的汉诺塔实现）使用栈而不是使用递归，实现程序清单 18-8 中的 moveDisks 方法。

**20.23 （表达式求值）修改程序清单 20-9，增加指数运算符 ^ 和求模运算符 %。例如，3^ 2 等于 9，3 % 2 等于 1。运算符 ^ 具有最高优先级，运算符 % 具有与 * 和 / 运算符一样的优先级。程序应该提示用户输入一个表达式。下面是一个程序的运行示例：

```
Enter an expression: (5 * 2 ^ 3 + 2 * 3 % 2) * 4 ↵Enter
(5 * 2 ^ 3 + 2 * 3 % 2) * 4 = 160
```