Exercise 2.34 Is the **increase** method a mutator? If so, how could you demonstrate this?

Exercise 2.35 Complete the following method, whose purpose is to subtract the value of its parameter from a field named **price**.

```
/**
 * Reduce price by the given amount.
 */
public void discount(int amount)
{
    ...
}
```

2.9 Printing from methods

Code 2.7 shows the most complex method of the class: **printTicket**. To help your understanding of the following discussion, make sure that you have called this method on a ticket machine. You should have seen something like the following printed in the BlueJ terminal window:

```
##################
# The BlueJ Line
# Ticket
# 500 cents.
##################
```

This is the longest method we have seen so far, so we shall break it down into more manageable pieces:

- The header indicates that the method has a **void** return type, and that it takes no parameters.
- The body comprises eight statements plus associated comments.
- The first six statements are responsible for printing what you see in the BlueJ terminal window: five lines of text and a sixth blank line.
- The seventh statement adds the balance inserted by the customer (through previous calls to **insertMoney**) to the running total of all money collected so far by the machine.
- The eighth statement resets the balance to zero with a basic assignment statement, in preparation for the next customer.

Code 2.7
The `printTicket` method

```
/**
 * Print a ticket.
 * Update the total collected and reduce the balance to zero.
 */
public void printTicket()
{
    // Simulate the printing of a ticket.
    System.out.println("##################");
    System.out.println("# The BlueJ Line");
    System.out.println("# Ticket");
    System.out.println("# " + price + " cents.");
    System.out.println("##################");
    System.out.println();

    // Update the total collected with the balance.
    total = total + balance;
    // Clear the balance.
    balance = 0;
}
```

Concept

The method **System.out.println** prints its parameter to the text terminal.

By comparing the output that appears with the statements that produced it, it is easy to see that a statement such as

```
System.out.println("# The BlueJ Line");
```

literally prints the string that appears between the matching pair of double-quote characters. The basic form of a call to `println` is

`System.out.println(`***something-we-want-to-print***`);`

where *something-we-want-to-print* can be replaced by any arbitrary string, enclosed between a pair of double-quote characters. For instance, there is nothing significant in the "#" character that is in the string—it is simply one of the characters we wish to be printed.

All of the printing statements in the **`printTicket`** method are calls to the **`println`** method of the **`System.out`** object that is built into the Java language, and what appears between the round brackets is the parameter to each method call, as you might expect. However, in the fourth statement, the actual parameter to **`println`** is a little more complicated and requires some more explanation:

```
System.out.println("# " + price + " cents.");
```

What it does is print out the price of the ticket, with some extra characters on either side of the amount. The two "+" operators are being used to construct a single actual parameter, in the form of a string, from three separate components:

- the string literal: **`"#  "`** (*note the space character after the hash*);
- the value of the **`price`** field (note that there are no quotes around the field name because we want the field's value, not its name);
- the string literal: **`" cents."`** (note the space character before the word **`"cents"`**).

When used between a string and anything else, "+" is a string-concatenation operator (i.e., it concatenates or joins strings together to create a new string) rather than an arithmetic-addition operator. So the numeric value of price is converted into a string and joined to its two surrounding strings.

Note that the final call to **`println`** contains no string parameter. This is allowed, and the result of calling it will be to leave a blank line between this output and any that follows after. You will easily see the blank line if you print a second ticket.

Exercise 2.36 Write down exactly what will be printed by the following statement:

```
System.out.println("My cat has green eyes.");
```

Exercise 2.37 Add a method called **`prompt`** to the **`TicketMachine`** class. This should have a **`void`** return type and take no parameters. The body of the method should print the following single line of output:

```
Please insert the correct amount of money.
```

Exercise 2.38 What do you think would be printed if you altered the fourth statement of **`printTicket`** so that **`price`** also has quotes around it, as follows?

```
System.out.println("# " + "price" + " cents.");
```

Exercise 2.39 What about the following version?

```
System.out.println("# price cents.");
```

Exercise 2.40 Could either of the previous two versions be used to show the price of tickets in different ticket machines? Explain your answer.

Exercise 2.41 Add a **`showPrice`** method to the **`TicketMachine`** class. This should have a `void` return type and take no parameters. The body of the method should print:

```
The price of a ticket is xyz cents.
```

where ***`xyz`*** should be replaced by the value held in the **`price`** field when the method is called.

Exercise 2.42 Create two ticket machines with differently priced tickets. Do calls to their **`showPrice`** methods show the same output, or different? How do you explain this effect?