

Chapter 2

Java Programming Basics

OBJECTIVES

After you have read and studied this chapter, you should be able to

- Identify the basic components of Java programs.
- Distinguish two types of Java programs—applications and applets.
- Write simple Java applications and applets.
- Describe the difference between object declaration and object creation.
- Describe the process of creating and running Java programs.
- Use `MainWindow` and `MessageBox` classes from the `javabook` package to write Java applications.
- Use the `Graphics` class from the standard Java package.

```
/*
Program MyFirstApplication

This program displays a window on the screen. The window is
positioned at the center of the screen, and the size of the
window is almost as big as the screen.
*/

import javabook.*;

class MyFirstApplication
{
    public static void main(String args[])
    {
        MainWindow mainWindow;
        mainWindow = new MainWindow();
        mainWindow.show();
    }
}
```

FIGURE 2.1 Result of running the **MyFirstApplication** program. The window has a default title **Sample Java Application**. (We'll show you how to change the default title later.)



FIGURE 2.2 The object diagram for the **MyFirstApplication** program.

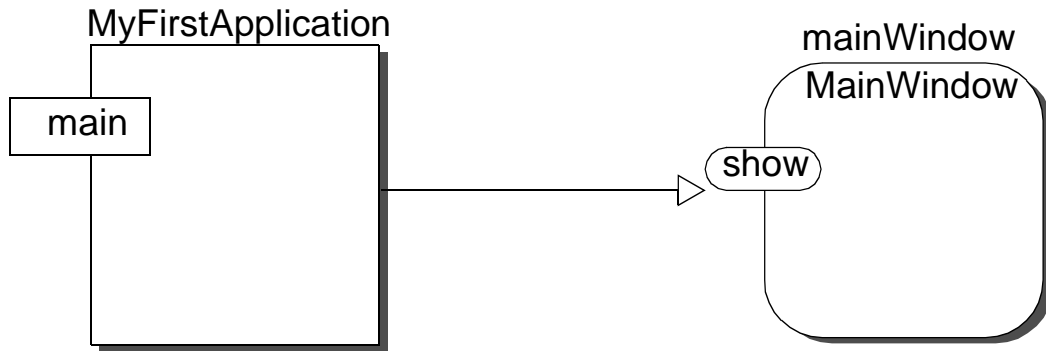


FIGURE 2.3 Result of running the program **MyFirstApplication** when the argument to the `new` command is the string argument **"This is my first window"**.



FIGURE 2.4 Distinction between object declaration and object creation.

State of Memory

Ⓐ `Account account;`

`account = new Account();`

The identifier **account** is declared and placed in memory.

after Ⓐ is executed

account 

Ⓑ `account = new Account();`

An **Account** object is created and the identifier **account** is set to refer to it.

after Ⓑ is executed

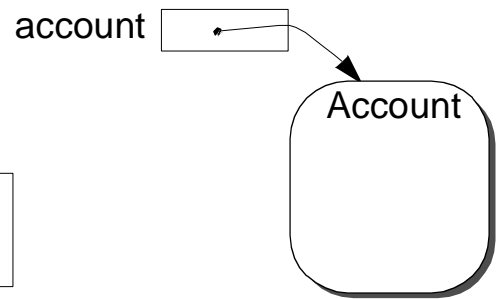
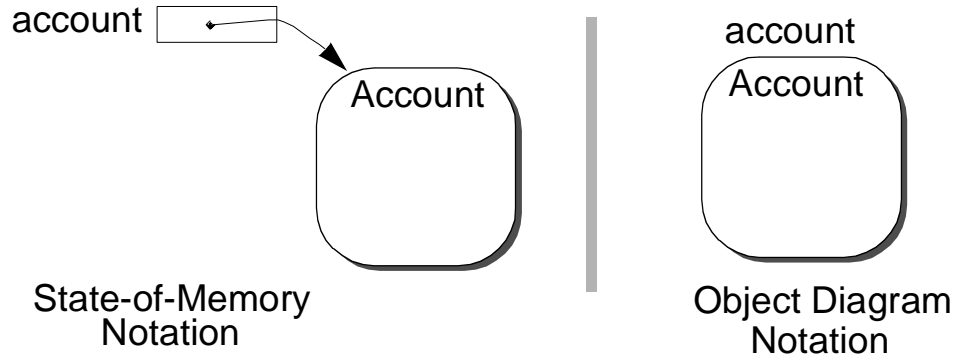


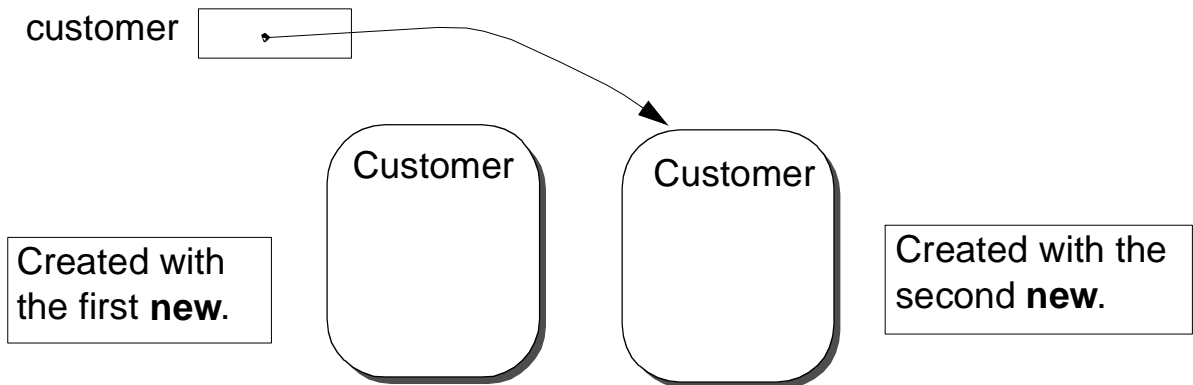
FIGURE 2.5 Relationship between the state-of-memory diagram and the object diagram notation.



The notation we use in object diagrams is a simplified version of the one used in state-of-memory diagrams.

FIGURE 2.6 The state after two **new** commands are executed.

```
Customer customer;
customer = new Customer();
customer = new Customer();
```



The first **Customer** object will be deallocated eventually because there are no references to it anymore.

FIGURE 2.7 Correspondence between message sending as represented in the object diagram and in the actual Java statement.

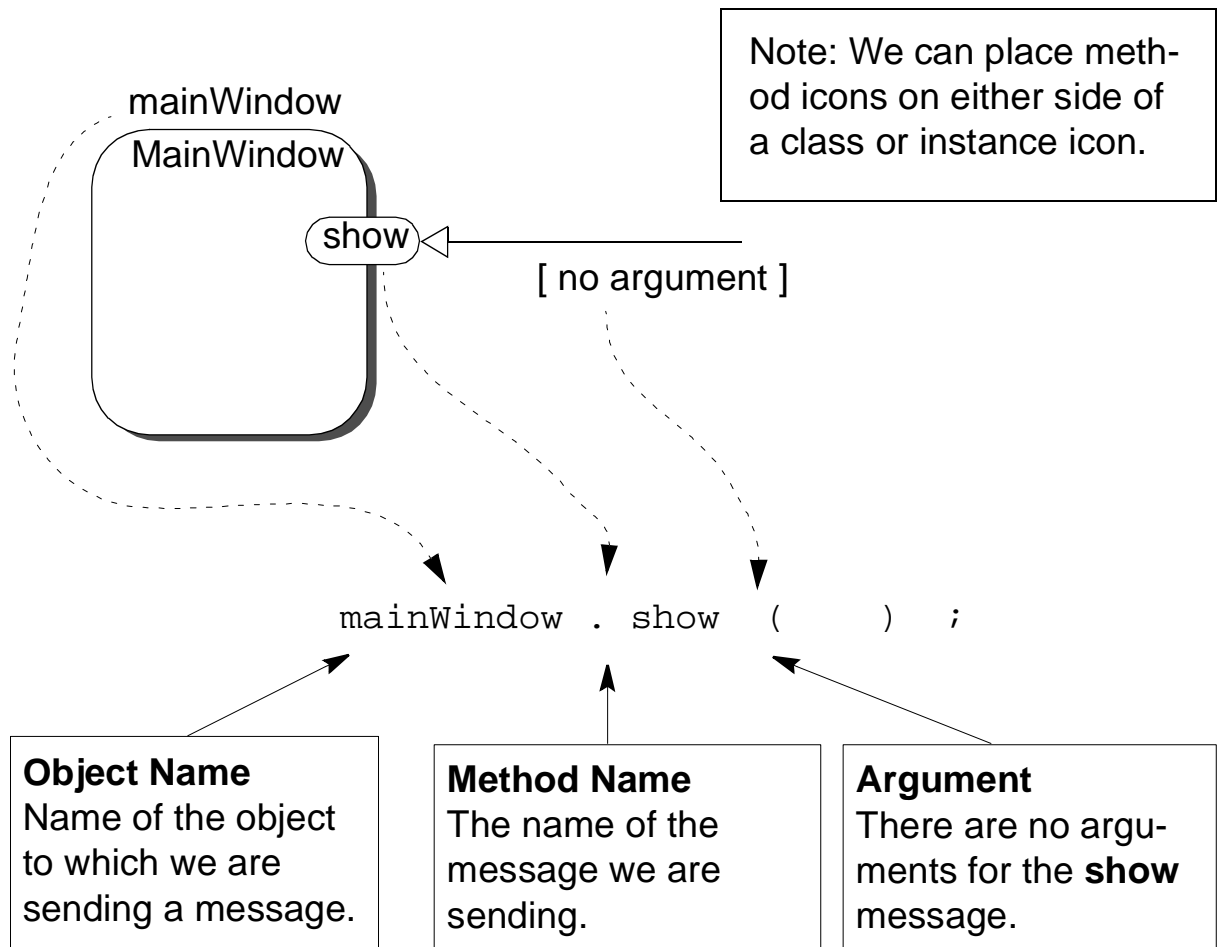
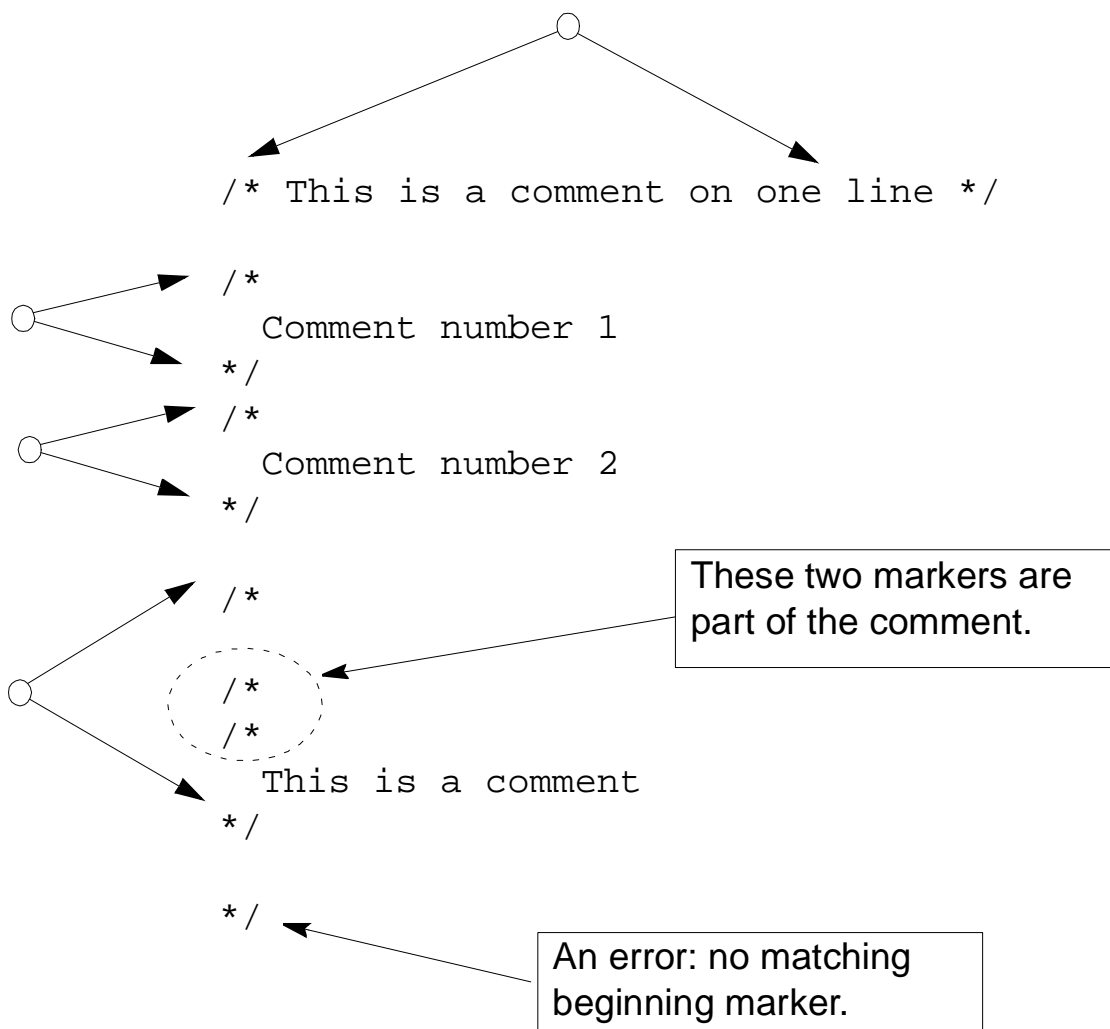


FIGURE 2.8 How the beginning and ending comment markers are matched.



```

/*****

Program: TextEditor

Author: C. Thomas Wu
       ctwu@cs.nps.navy.mil

Written: January 1, 1998

Course: Comp Sci 101
       Spring 98
       Program Assignment No. 7

Compiler:JDK 1.1.5
Platform:Windows 95

Description:
  This is a simple text editor. The editor allows
  the user to save text to a file and read text from
  a file. The editor displays text using Courier
  font only and does not allow formatting (e.g.,
  bold, italic, etc.). The editor supports standard
  editing functions Cut, Copy, and Paste, but does
  not support Undo. For more details, please refer
  to the TxEditReadme file.

*****/

```


FIGURE 2.9 A program template for simple Java applications.

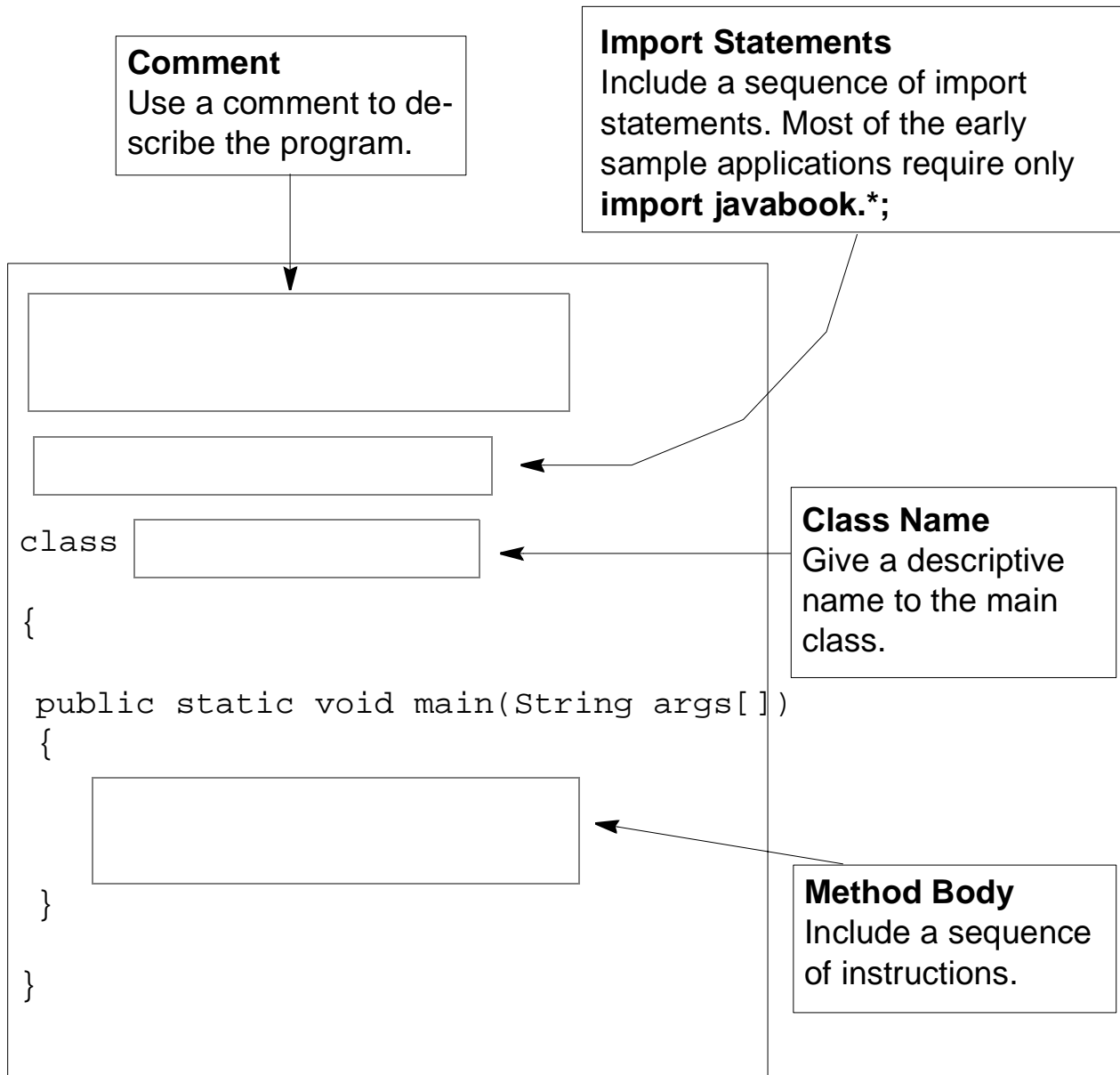


FIGURE 2.10 Result of running the **DisplayMessage** program.

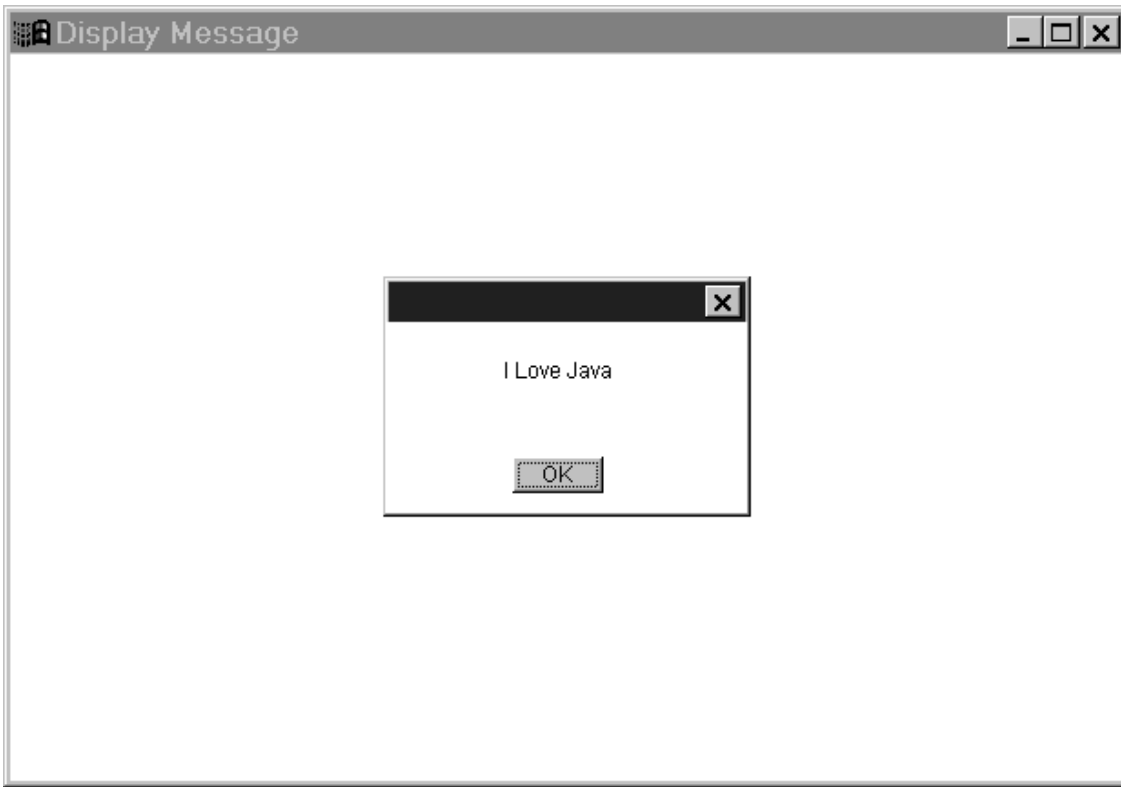


FIGURE 2.11 The object diagram for the **DisplayMessage** program.

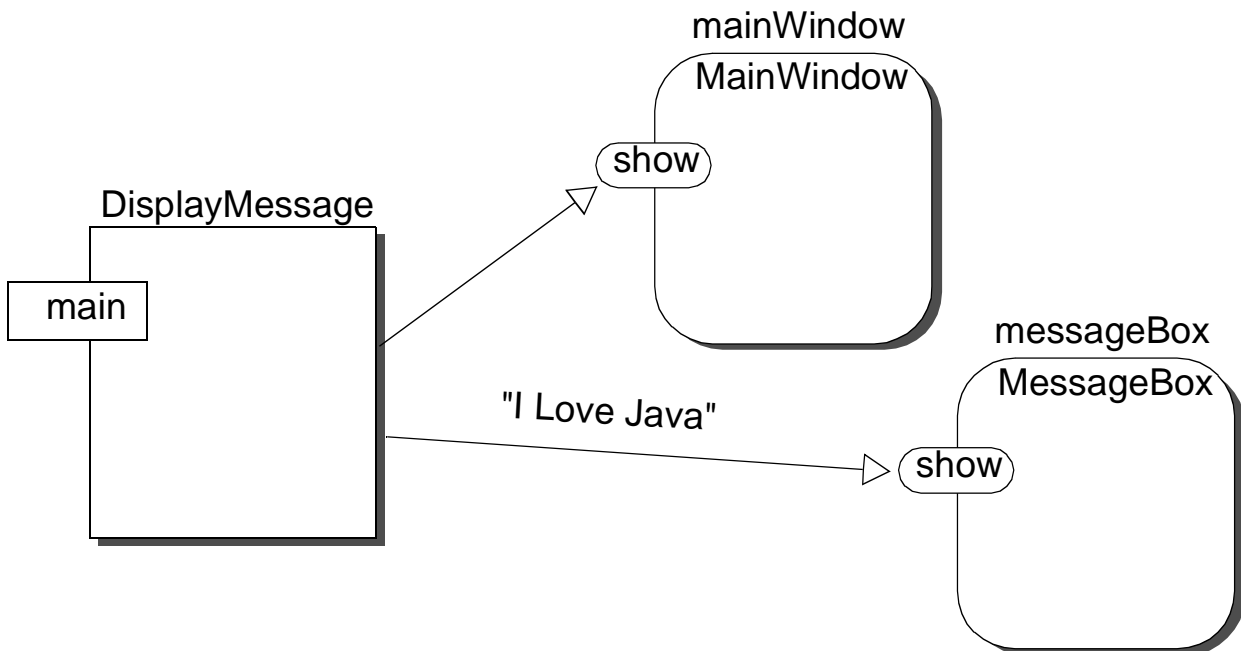


FIGURE 2.12 The applet **MyFirstApplet** with the three program components identified.

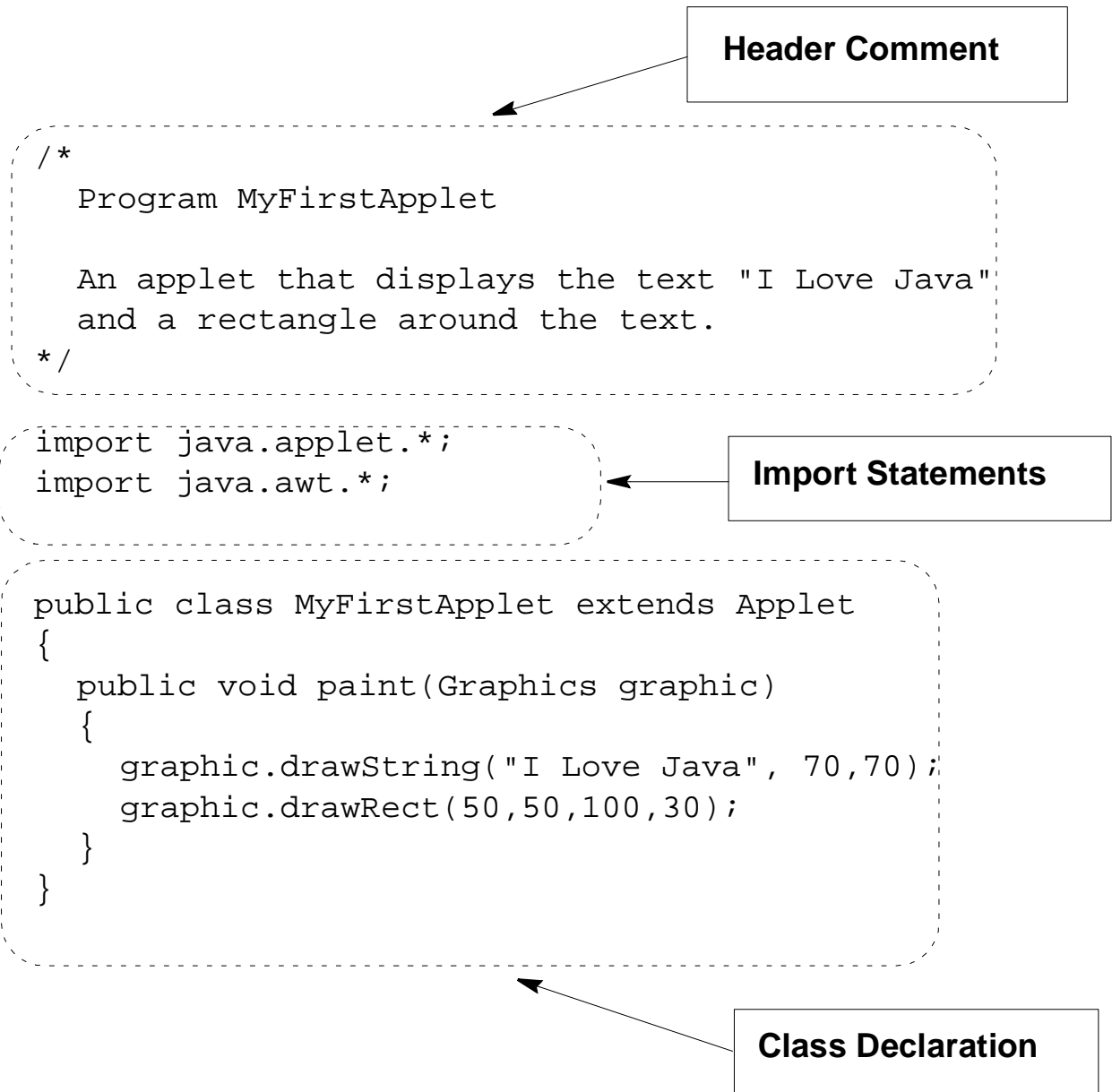


FIGURE 2.13 The object diagram for the **MyFirstApplet** program.

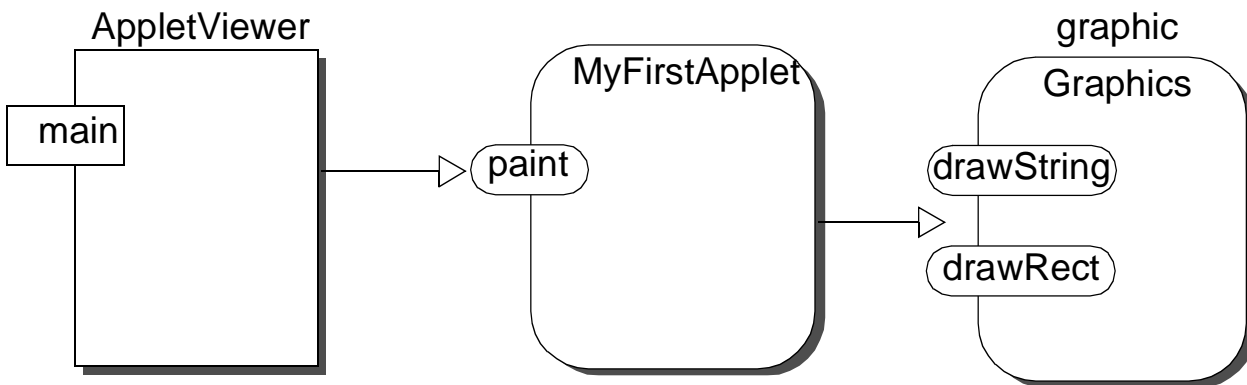


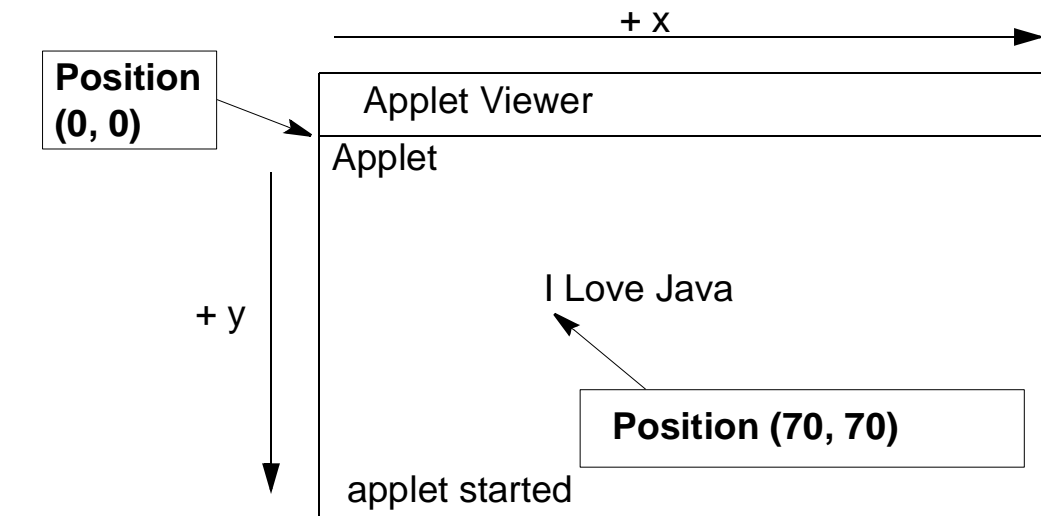
FIGURE 2.14 The diagram illustrates how the position of text is determined by the **drawString** method.

Syntax

<text> is displayed at position (<x>,<y>).

```
graphic.drawString( <text>, <x>, <y>);
```

Example: `graphic.drawString("I Love Java", 70,70);`



Position (70, 70)
specifies the lower-left corner of text.

I Love Java

Close-up View

FIGURE 2.15 The diagram illustrates how the position of a rectangle is determined by the **drawRect** method.

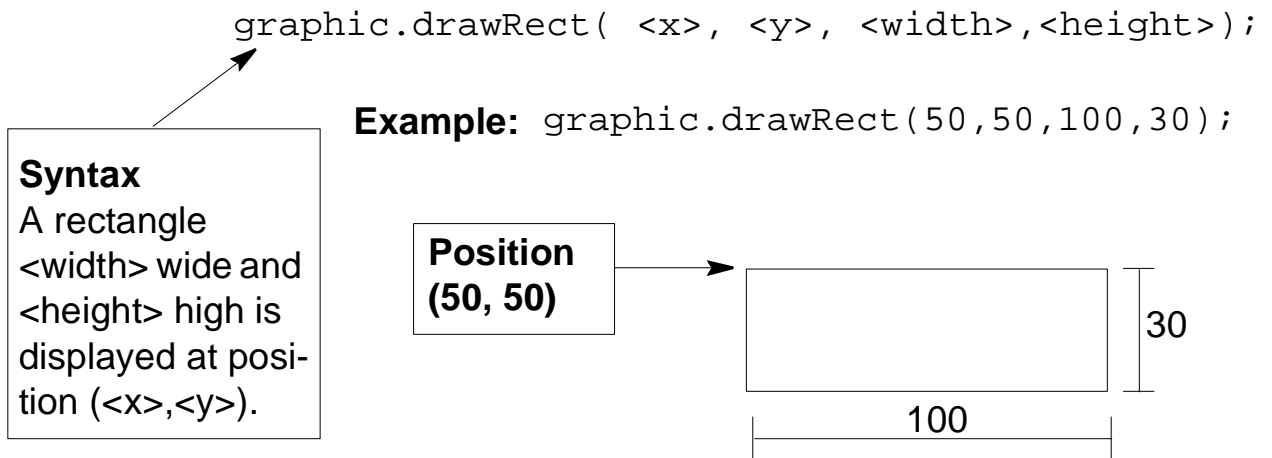


TABLE 2.1 A partial list of drawing methods defined for the **Graphics** class.

Method	Meaning
<code>drawLine(x1, y1, x2, y2)</code>	Draws a line between (x1,y1) and (x1,y2).
<code>drawRect(x, y, w, h)</code>	Draws a rectangle with width w and height h at (x,y).

FIGURE 2.16 A program template for simple Java applets.

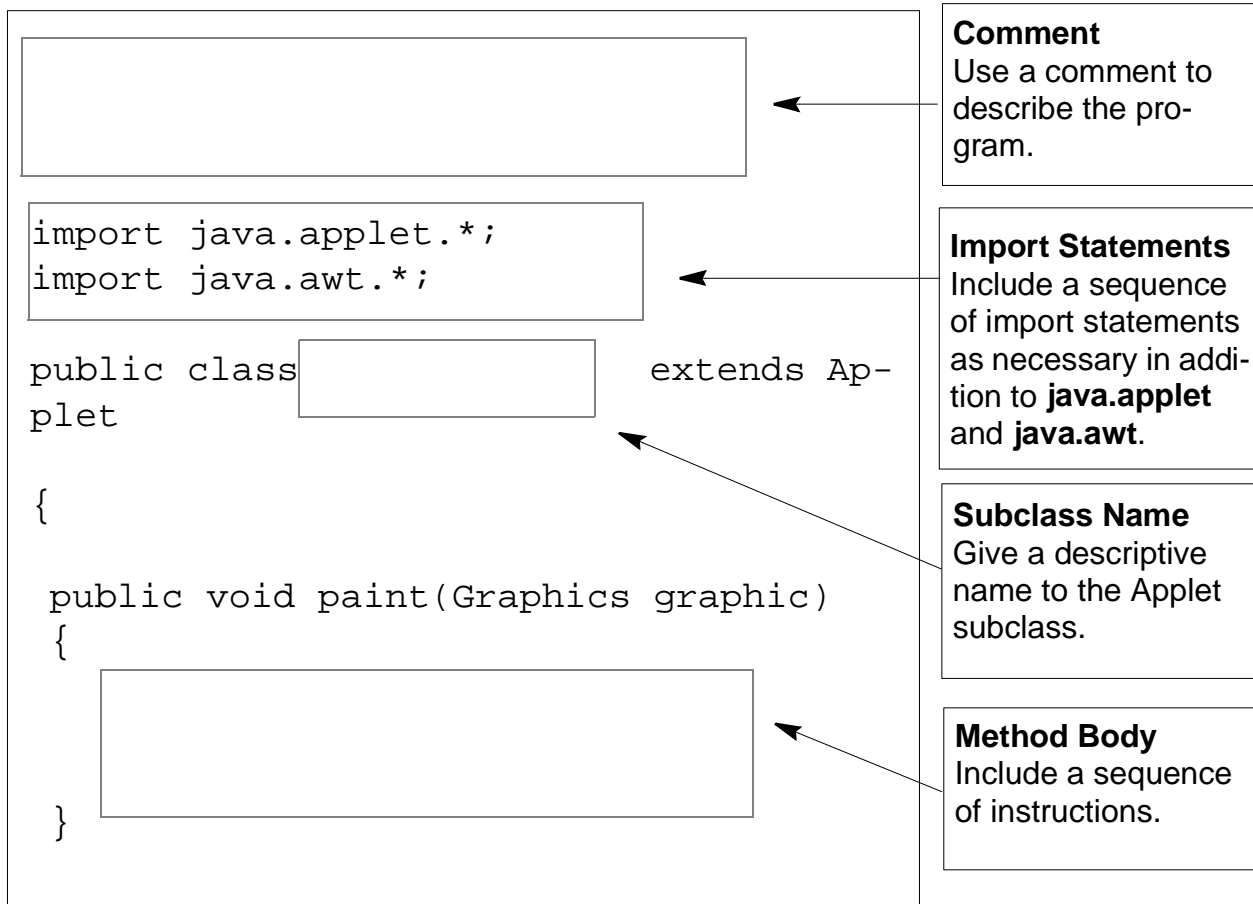


TABLE 2.1 (Continued) A partial list of drawing methods defined for the

Method	Meaning
<code>drawRoundRect</code> <code>(x,y,w,h,aw,ah)</code>	Draws a rounded-corner rectangle with width <i>w</i> and height <i>h</i> at (x,y) . Parameters <i>aw</i> and <i>ah</i> determine the angle for the rounded corners.

TABLE 2.1 (Continued) A partial list of drawing methods defined for the

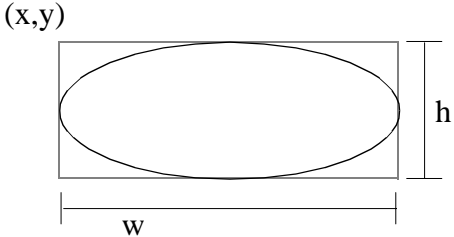
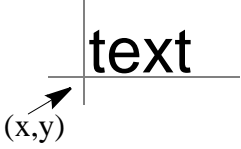
Method	Meaning
<code>drawOval(x, y, w, h)</code>	<p>Draws an oval with width w and height h at (x,y).</p> 
<code>drawString("text", x, y)</code>	<p>Draws the string <code>text</code> at (x,y).</p> 

FIGURE 2.17 Result of running the applet **MyFirstApplet**.

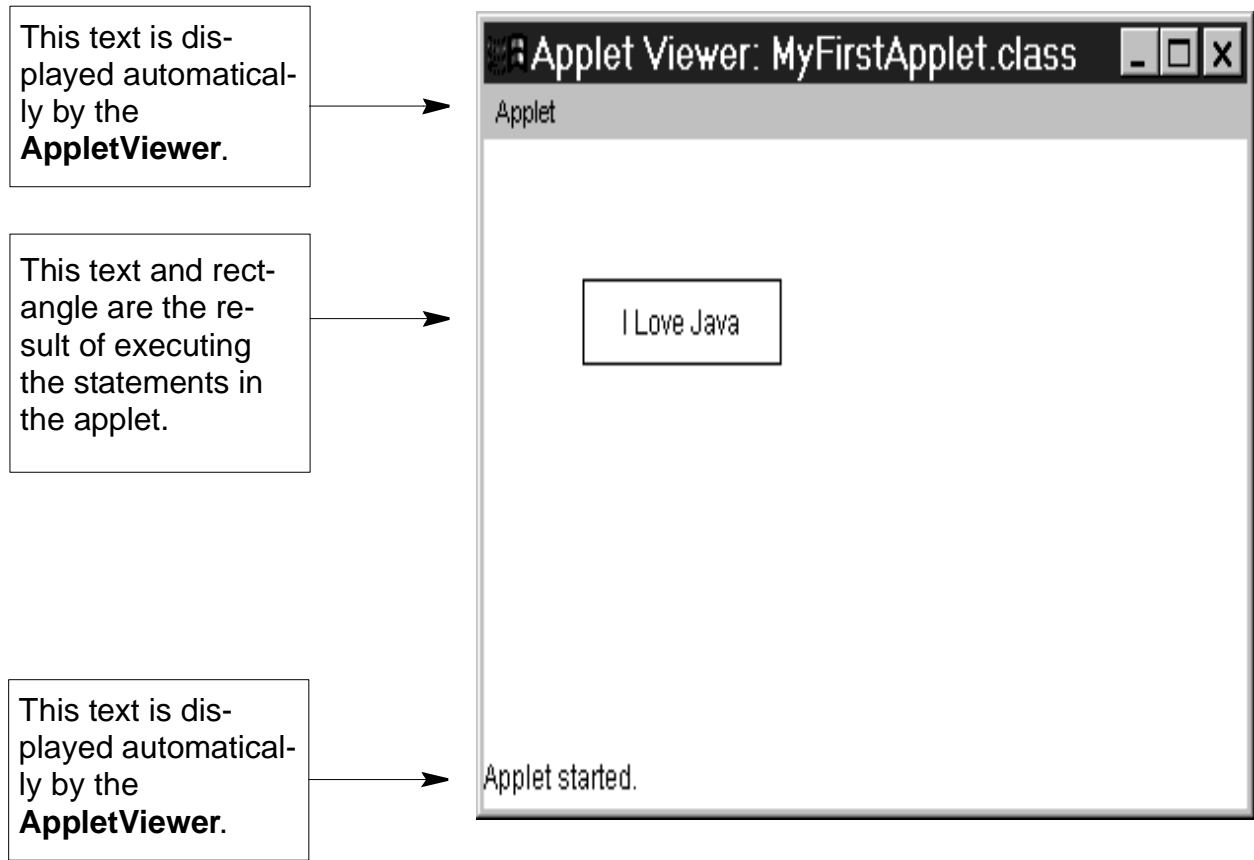


FIGURE 2.18 Result of running the applet **MyFirstApplet** in Netscape Navigator.

