Chapter 2

Java Programming Basics

Chapter 2 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.

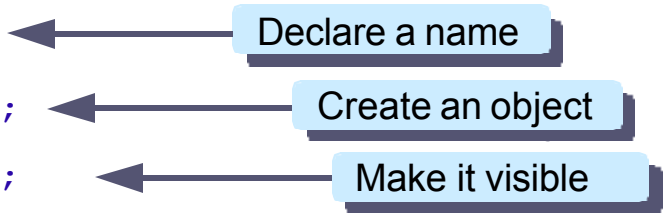
The First Java Application

- A program to display a window on the screen.
- The size of the window is slightly smaller than the screen, and the window is positioned at the center of the screen with a default title Sample Java Application.
- The fundamental OOP concept illustrated by the program:

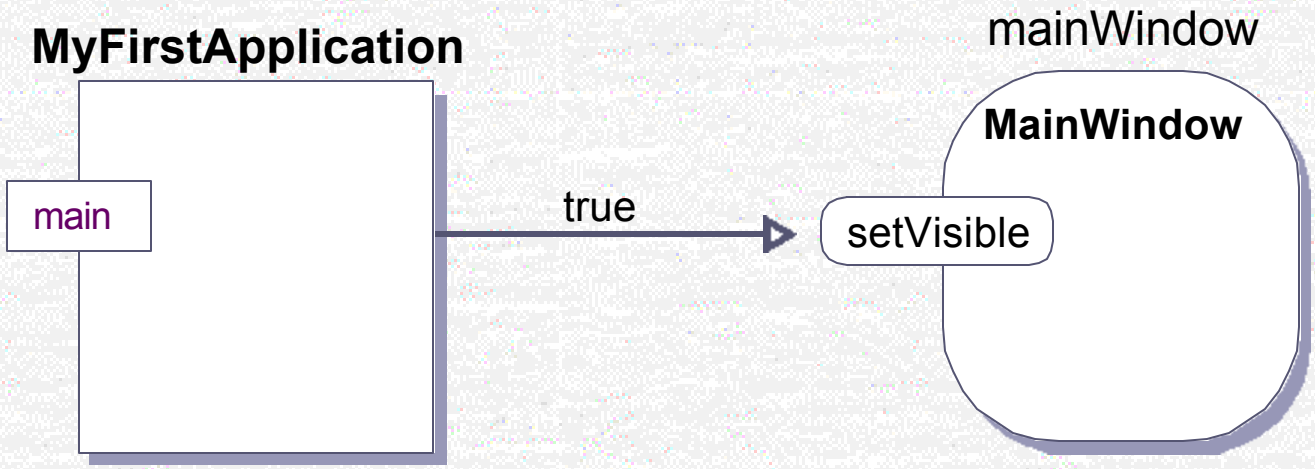
An object-oriented program uses objects.

Program MyFirstApplication

```
/*
   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.setVisible( true );
   }
}
```

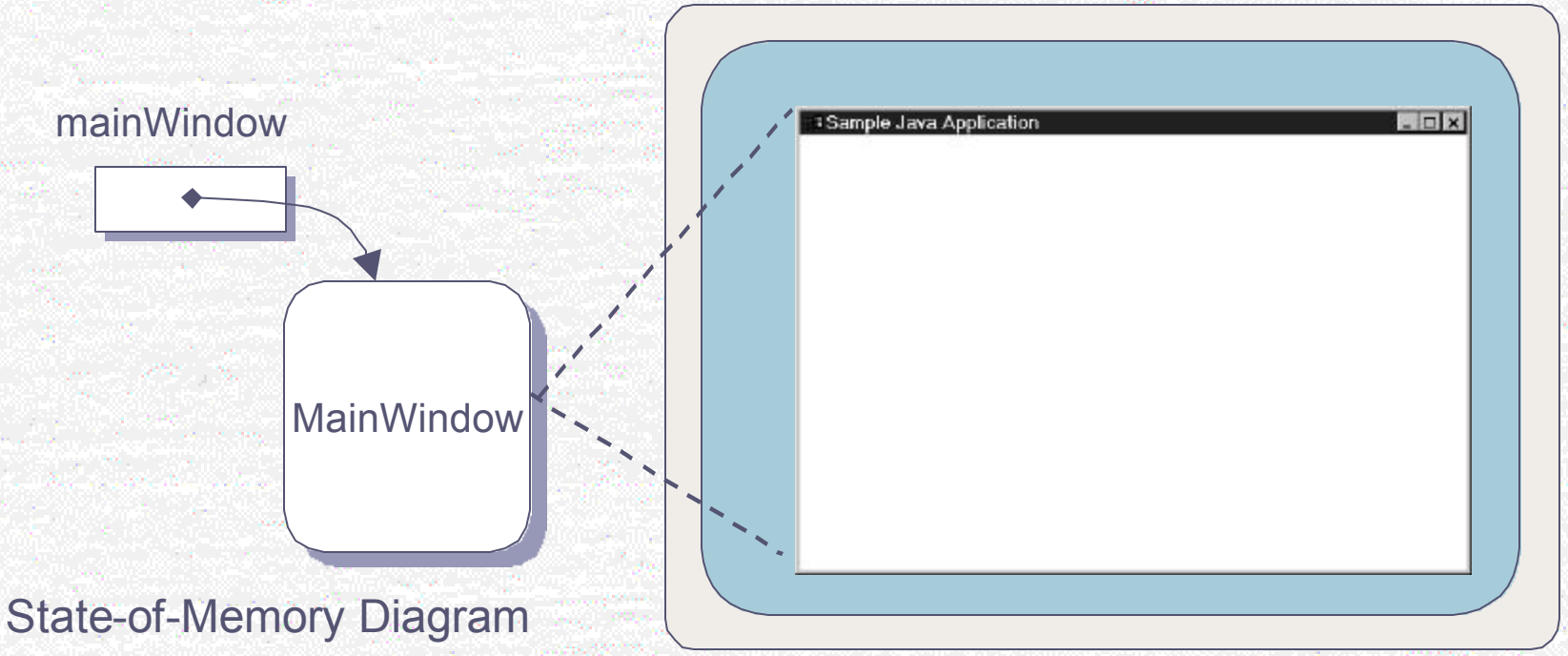


Object Diagram for MyFirstApplication



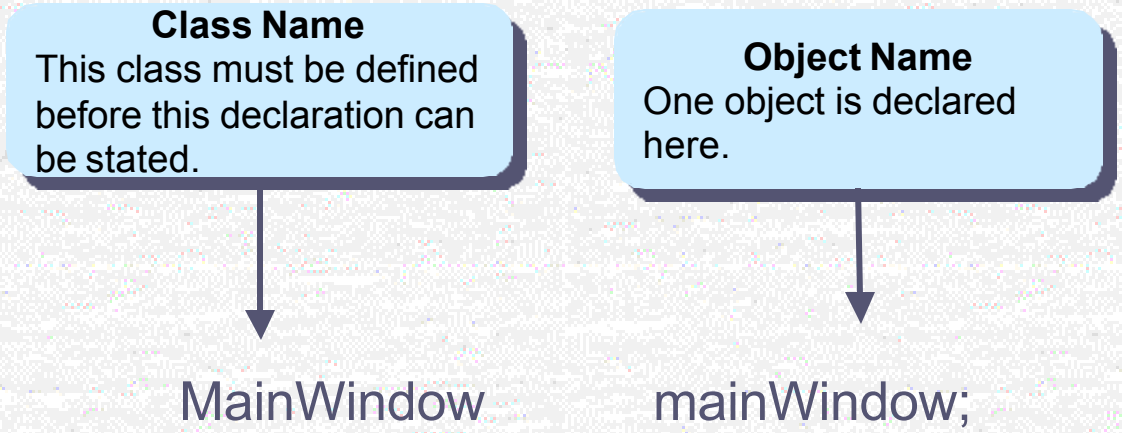
Flow of the MyFirstApplication Program

```
MainWindow  mainWindow;
mainWindow = new MainWindow();
mainWindow.setVisible( true );
```



State-of-Memory Diagram

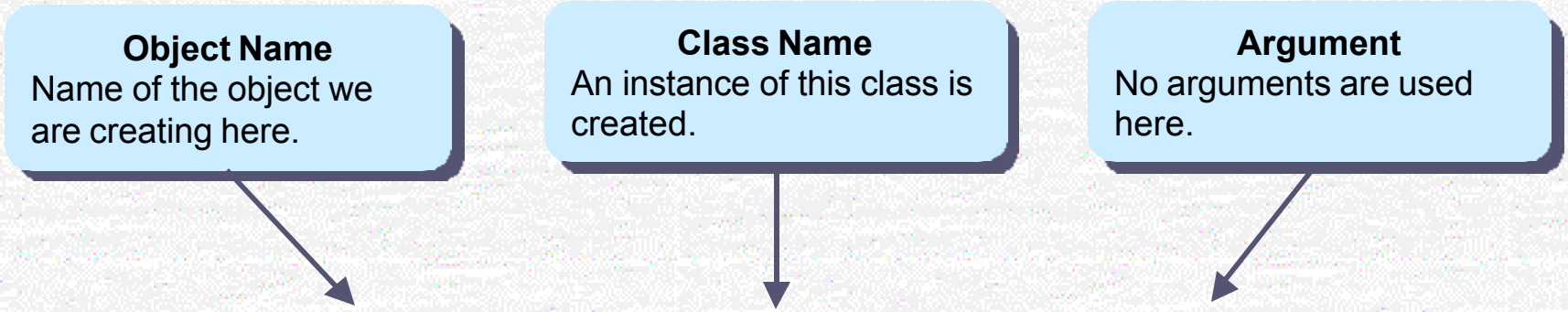
Object Declaration



More Examples

```
Account     customer;
Student     jan, jim, jon;
Vehicle     car1, car2;
```

Object Creation

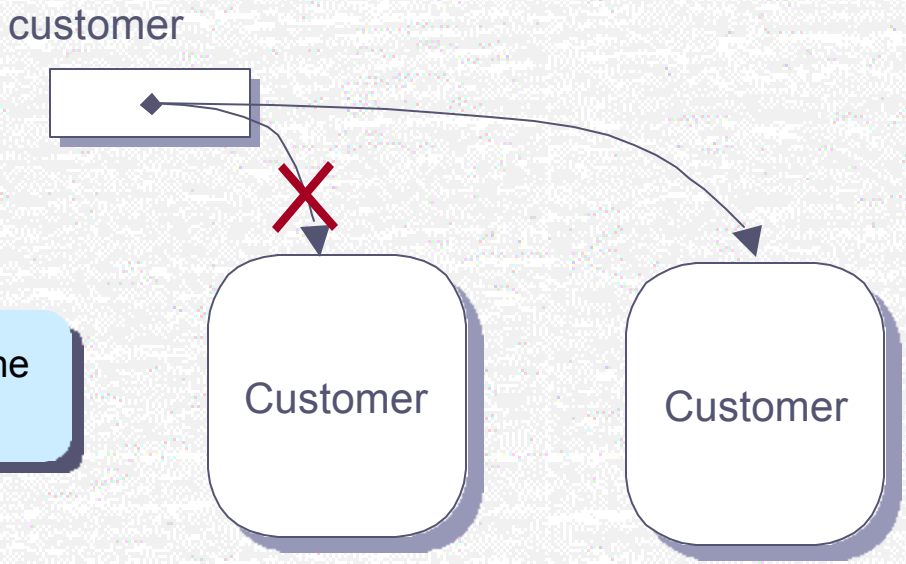


More Examples

```
customer = new Customer( );
jon      = new Student("John Java" );
car1     = new Vehicle( );
```

Distinction Between Declaration and Creation

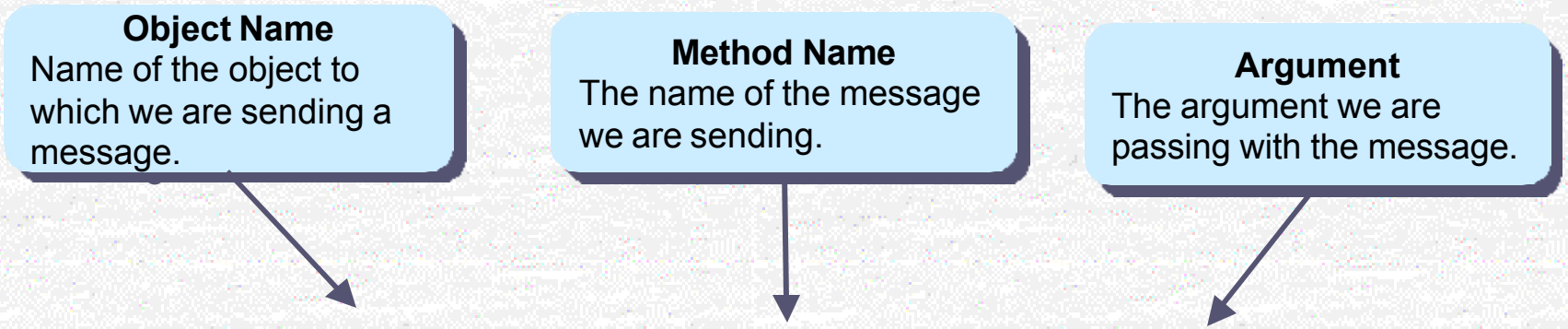
```
Customer   customer;
customer   =  new  Customer( );
customer   =  new  Customer( );
```



Created with the first **new**.

Created with the second **new**. Reference to the first Customer object is lost.

Sending a Message



More Examples

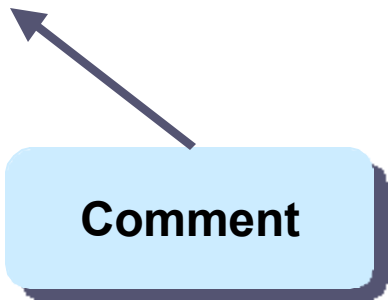
```
account.deposit( 200.0 );
student.setName("john");
car1.startEngine( );
```

Program Components

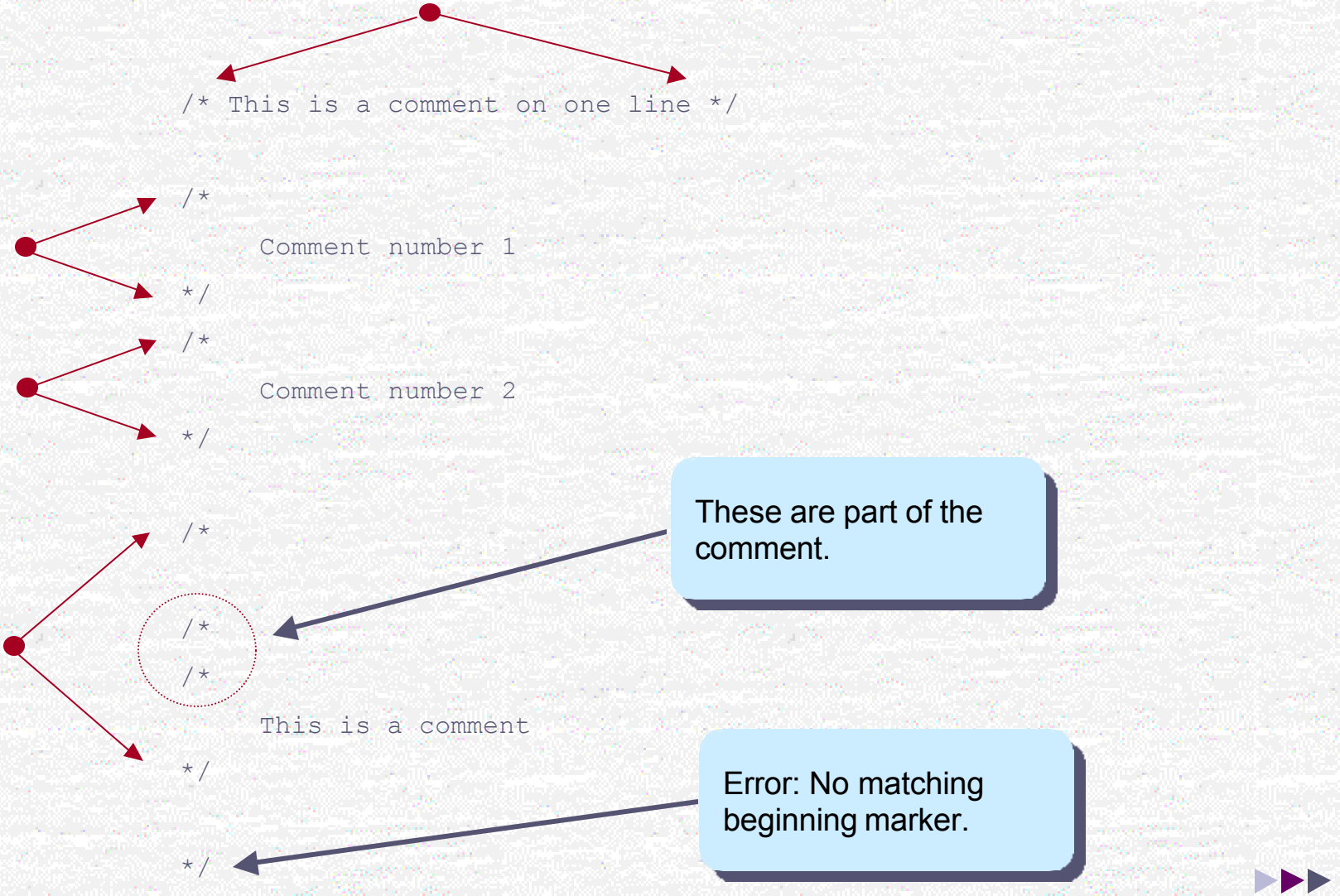
- A Java program is composed of
  - comments,
  - import statements, and
  - class declarations.

Program Component: Comment

```
/*
    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.setVisible( true );
    }
}
```



Matching Comment Markers



Three Types of Comments

```
/*
     This is a comment with
     three lines of
     text.
*/
```

Multiline Comment

```
// This is a comment
// This is another comment
// This is a third comment
```

Single line Comments

```
/**
 * This class provides basic clock functions. In addition
 * to reading the current time and today's date, you can
 * use this class for stopwatch functions.
 */
```

javadoc Comments

Program Component: Import Statement

```
/*
    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.setVisible( true );
    }
}
```

Import Statement Syntax and Semantics

Package Name
Name of the package that contains the classes we want to use.

Class Name
The name of the class we want to import. Use asterisks to import all classes.

<package name> . <class name> ;

e.g. javabook . InputBox;

More Examples

```
import    javabook.*;
import    java.awt.image.ColorModel;
import    com.drcaffeine.galapagos.*;
```

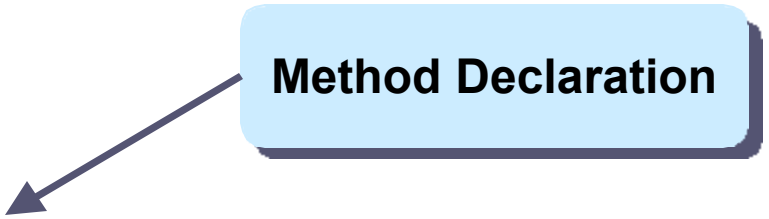
Program Component: Class Declaration

```
/*
    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.setVisible( true );
    }
}
```

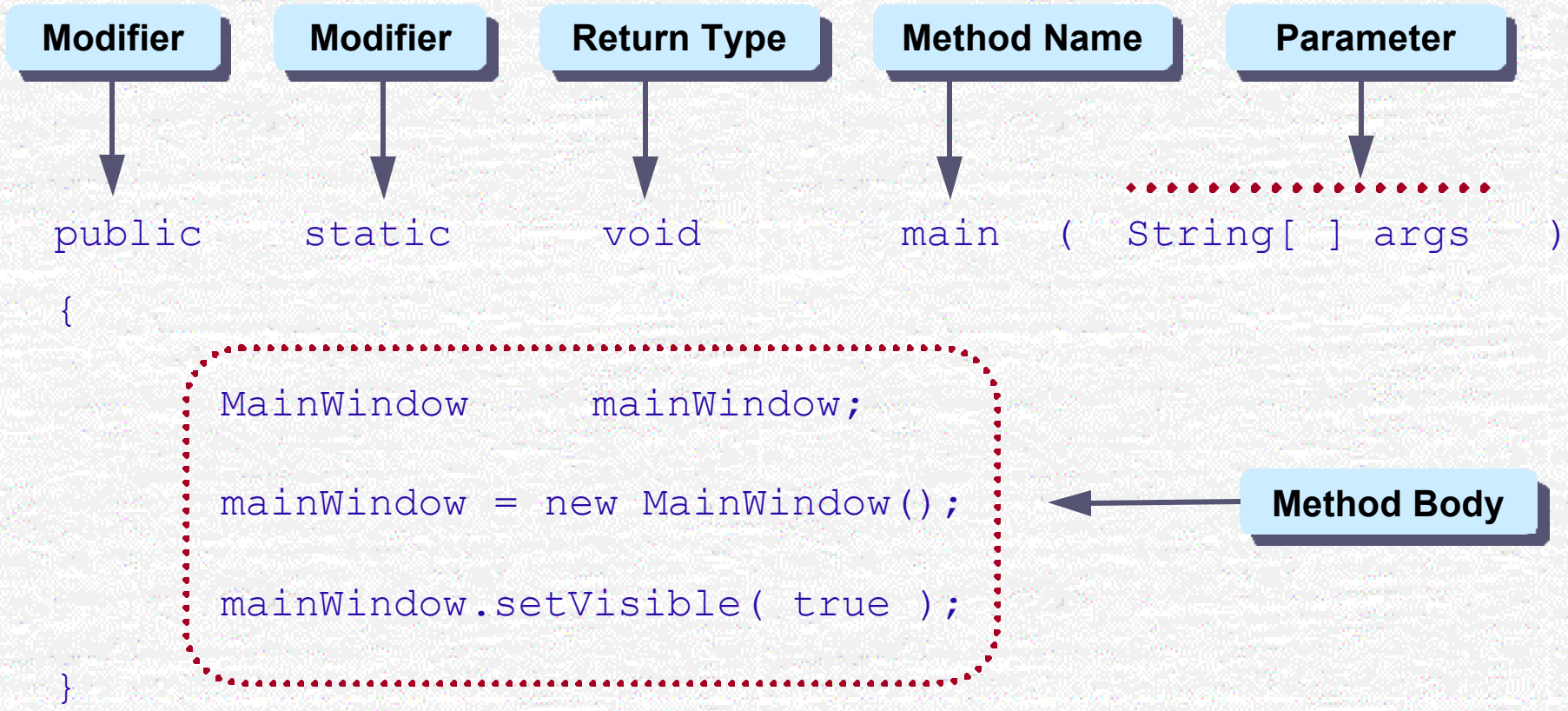
Class Declaration

Program Component: Method Declaration

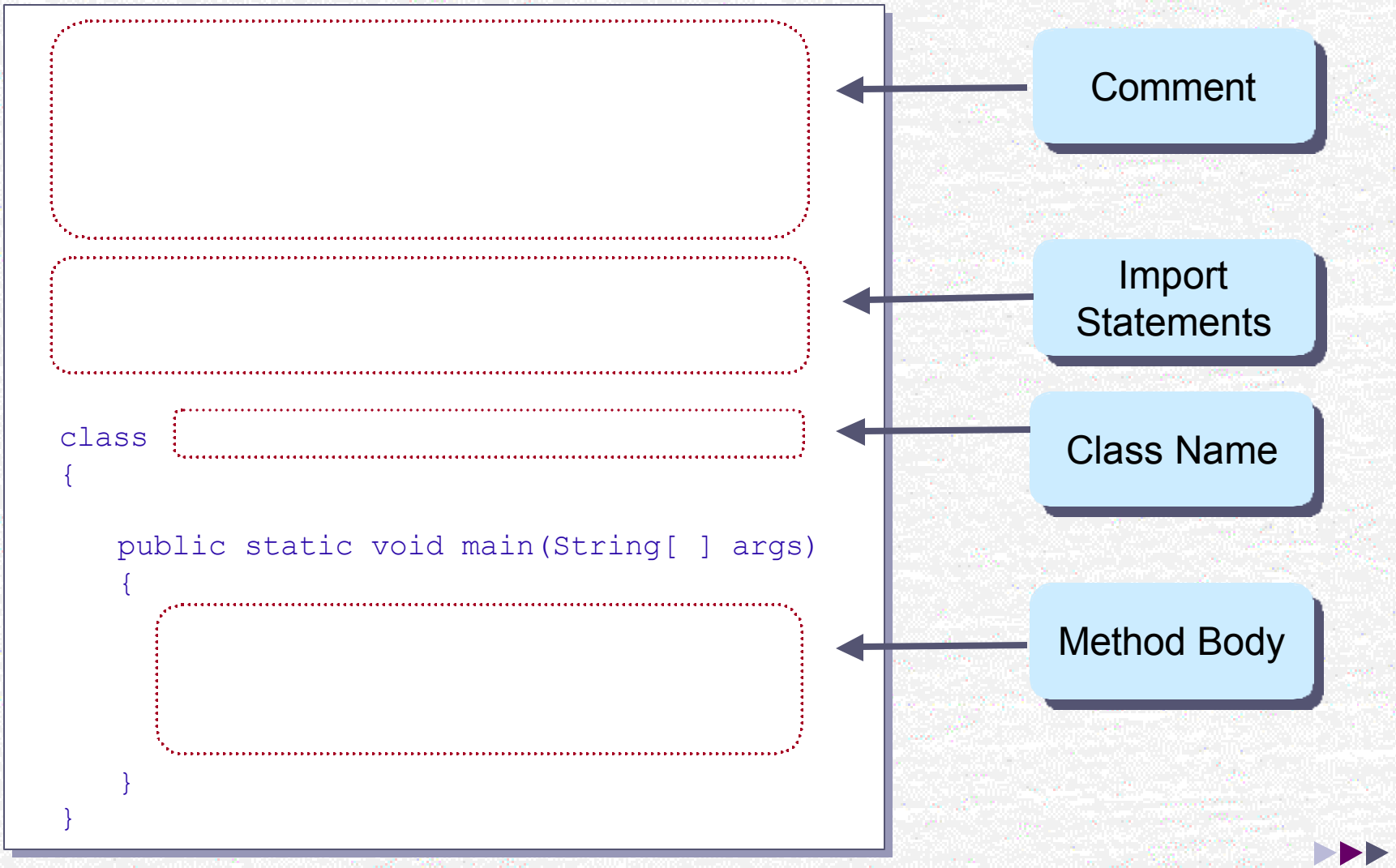
```
/*
    Program MyFirstApplication

    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.setVisible( true );
   }
}
```



Method Declaration Elements



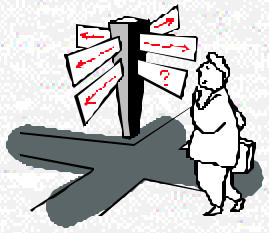
Template for Simple Java Applications



Steps in Executing Java Applications

- Step 1 Edit
  - Type in the program using an editor and save the program to a file.
- Step 2 Compile
  - Compile the source file.
- Step 3 Run
  - Execute the compiled source file called bytecode file.

Click this image to read step-by-step instructions on how to edit, compile, and run Java programs.



The javabook Package

- To become a good object-oriented programmer, one must first learn how to use predefined classes.
- We used predefined classes from the javabook package. To download the package or get its detailed documentation, please visit Dr. Caffeine's web site.
- Advantages of using javabook:
  - Gives you a taste of how real-world programs are developed.
  - Minimizes the impact of programming language syntax and semantics.
  - Allows you to write practical programs without learning too many details.
  - Serves as good example of how to design classes.

Sample Program: Displaying Messages

- Problem Statement

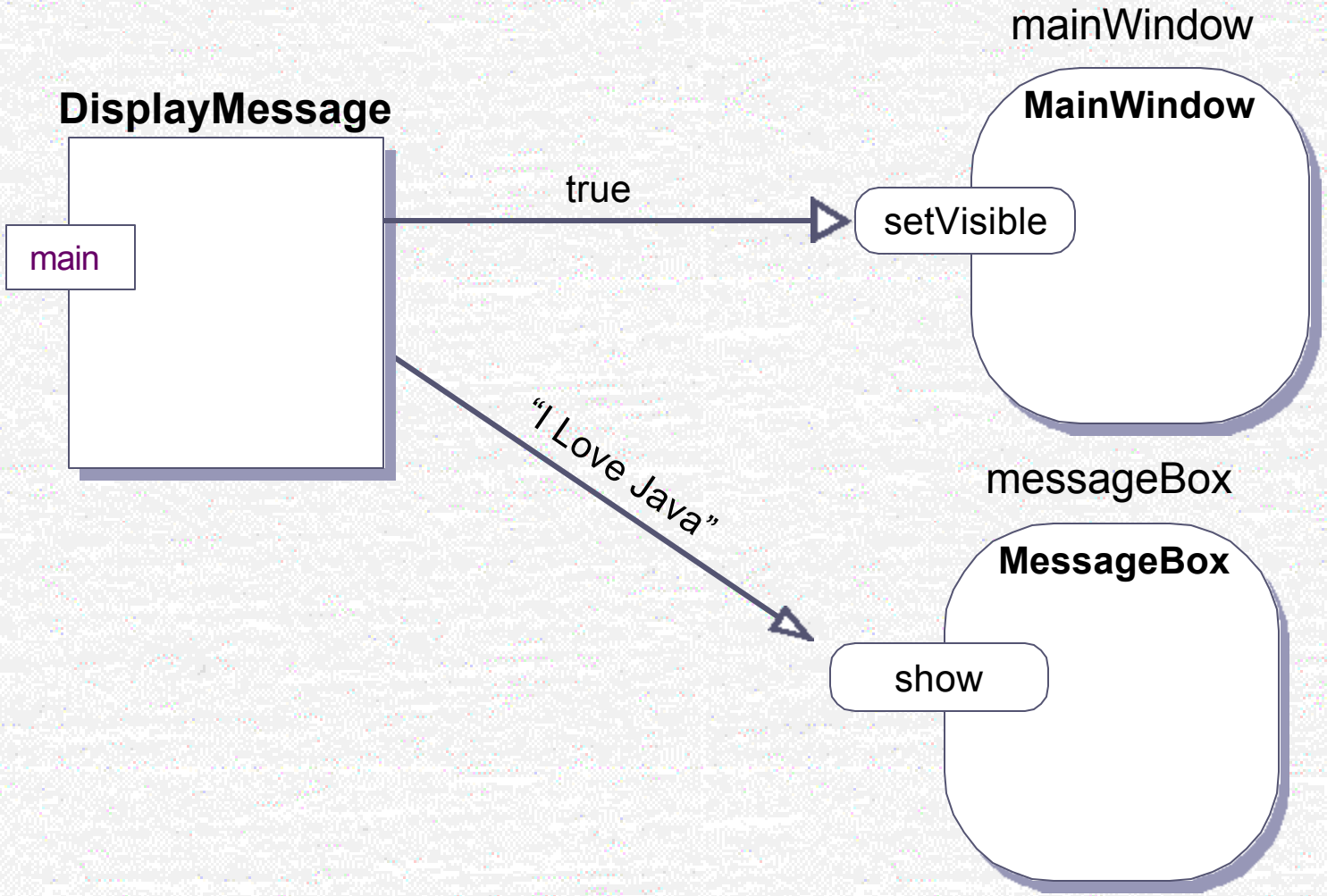
  Write an application that displays the message **I Love Java**.

- Design
  - **Alternative 1**: Set the title of the MainWindow to the designated message.
  - **Alternative 2**: Use a MessageBox object. This object is intended for displaying a single line of short text to grab the enduser's attention. The MessageBox class is available from the javabook package.

Sample Program: Design Document

Design Document: DisplayMessage Class	Purpose
DisplayMessage	The main class of theprogram.
MainWindow	The main frame window of the program. The title is set to Display Message. This class is from javabook.
MessageBox	The dialog for displaying the required message. This class is from javabook.

Sample Program: Object Diagram

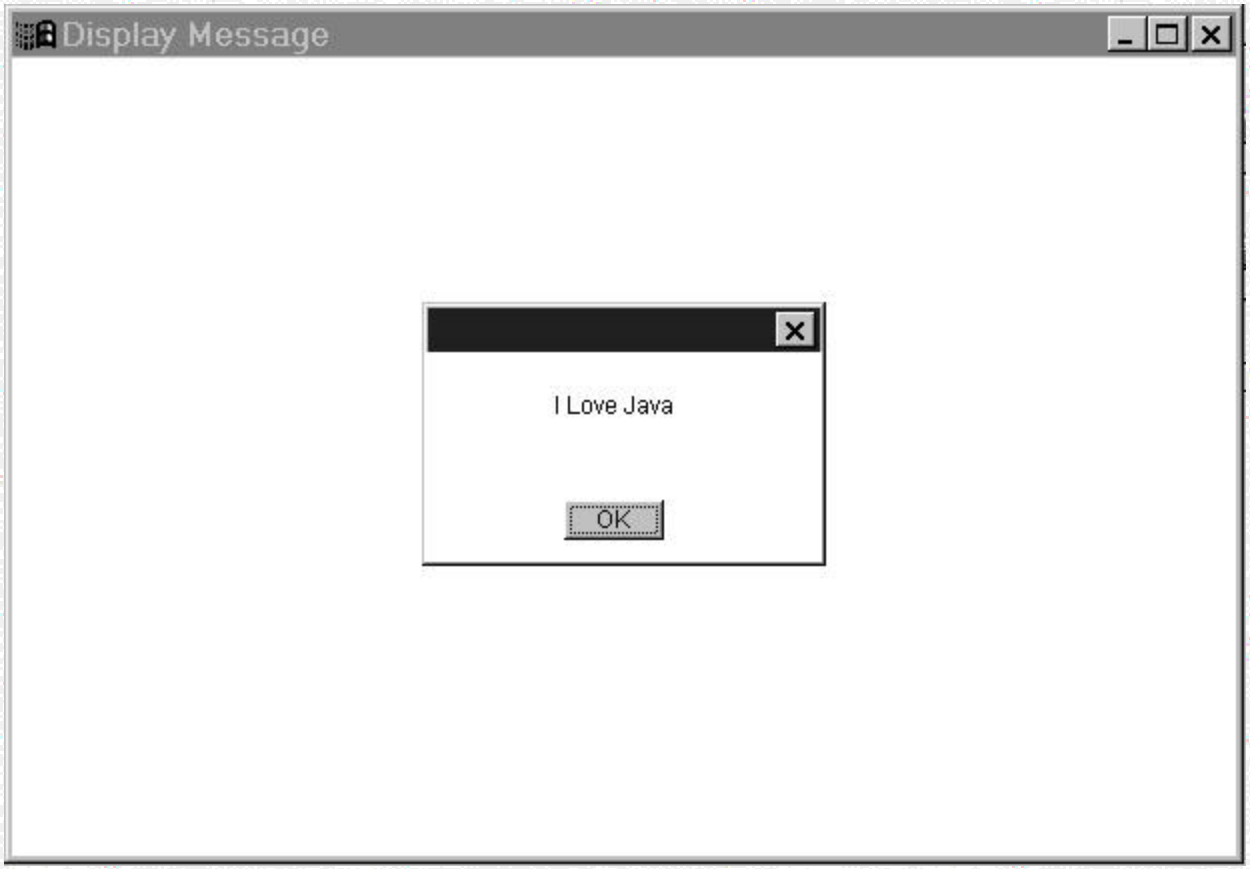


Sample Program: Source Code

```
/*
   Program DisplayMessage

   The program displays the text "I Love Java". The program uses a
   MessageBox object from the javabook package to display the text.
*/

import javabook.*;

class DisplayMessage
{
   public static void main(String[] args)
   {
       MainWindow mainWindow;            //declare two objects
       MessageBox messageBox;
                                         //create two objects
       mainWindow = new MainWindow("Display Message");
       messageBox = new MessageBox(mainWindow);

       mainWindow.setVisible( true );    //display two objects: first the frame
       messageBox.show("I Love Java");   //and then the dialog
   }
}
```

Sample Program: Testing

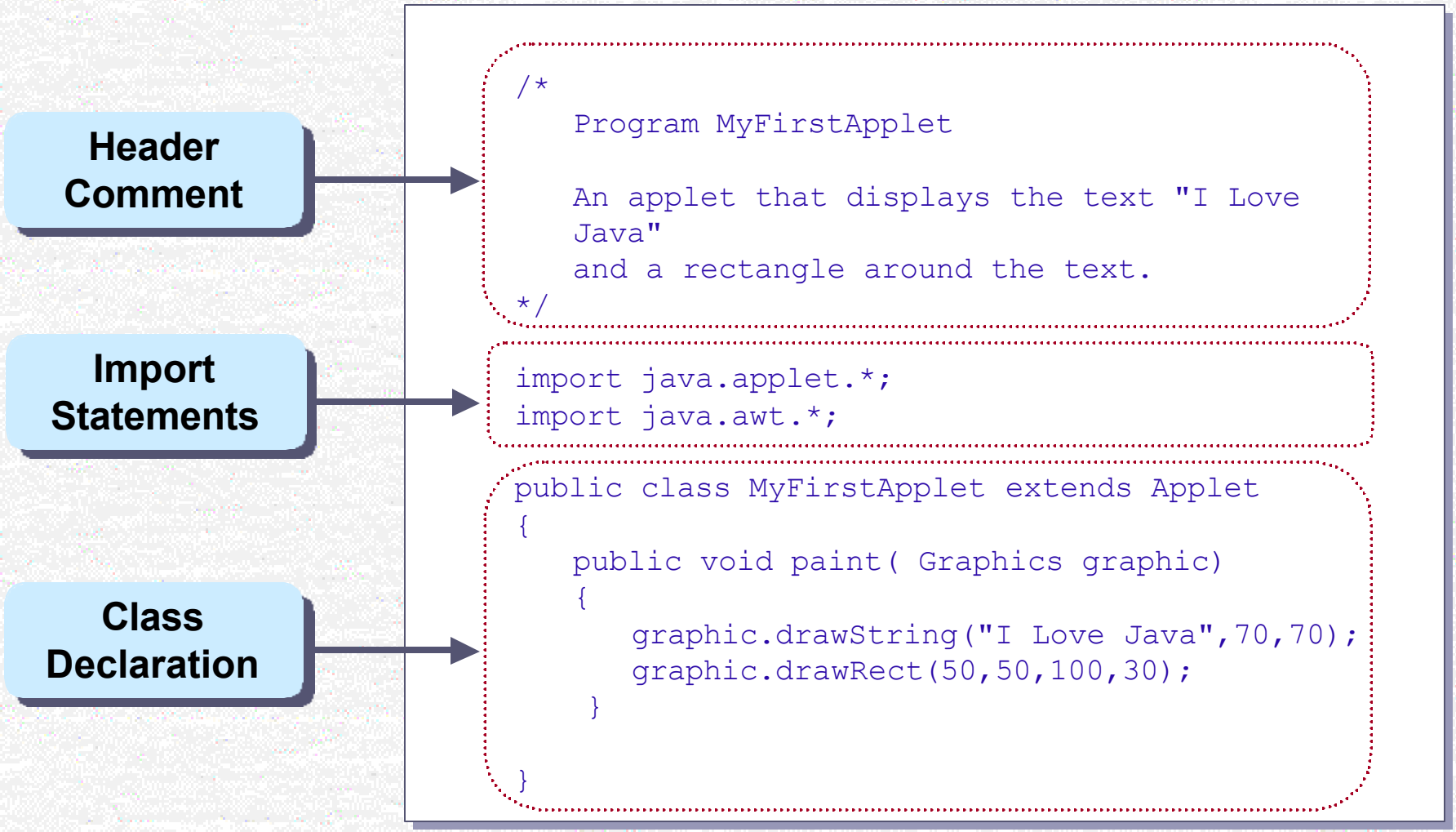
- Run the program, and you will see…



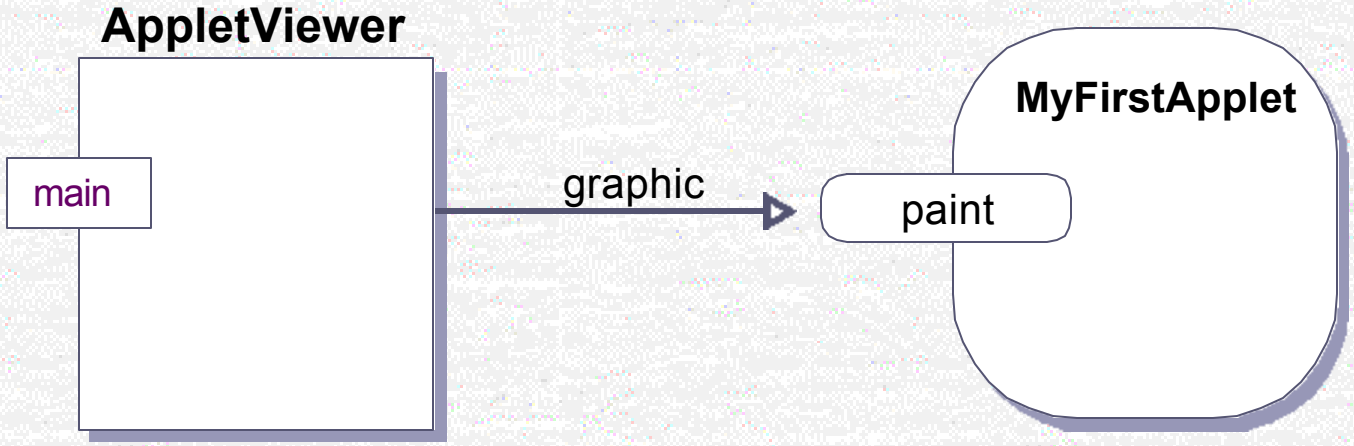
Program MyFirstApplet

```
/*
    Program MyFirstApplet

    An applet that displays the text "I Love Java"
    and a rectangle around the text.
*/

import java.applet.*;
import java.awt.*;

public class MyFirstApplet extends Applet
{
    public void paint( Graphics graphic)
    {
        graphic.drawString("I Love Java",70,70);
        graphic.drawRect(50,50,100,30);
     }

}
```

Three Components of Program MyFirstApplet



Object Diagram for MyFirstApplet



Drawing Graphics inside the paint Method

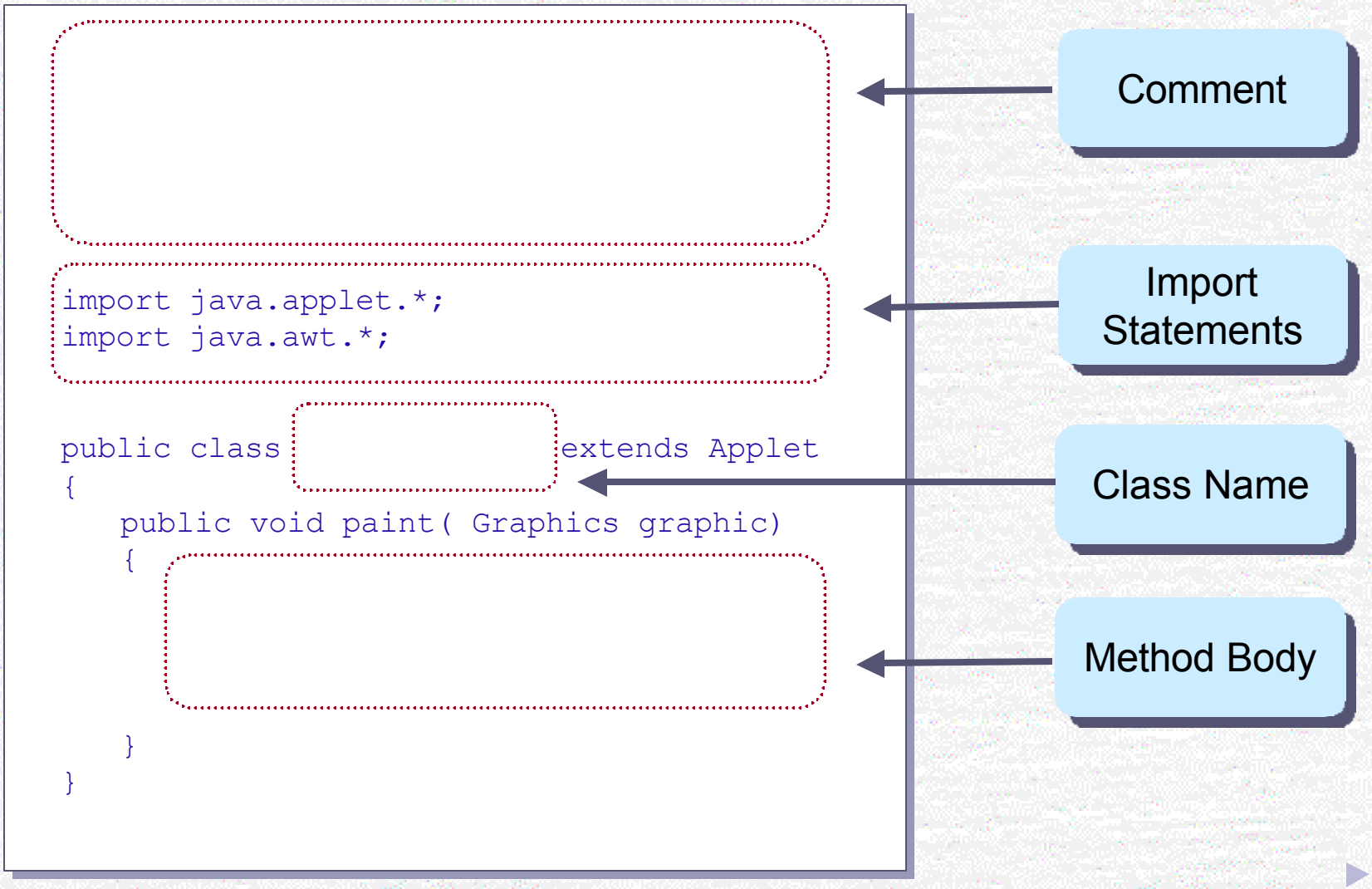
```
public void paint( Graphics graphic)
{
   graphic.drawString("I Love Java",70,70);
   graphic.drawRect(50,50,100,30);
}
```

Drawing
This is where we draw on an applet window by using the Graphics methods.

Drawing Methods

- drawLine(x1, y1, x2, y2)
  - draws a line from (x1,y1) to (x2, y2)
- drawRect(x, y, w, h)
  - draws a rectangle w pixels wide and h pixels high at (x,y).
- drawOval(x, y, w, h)
  - draws an oval w pixels wide and h pixels high at (x, y).
- See java.awt.Graphics for information on these and other drawing methods.

Template for Simple Java Applets

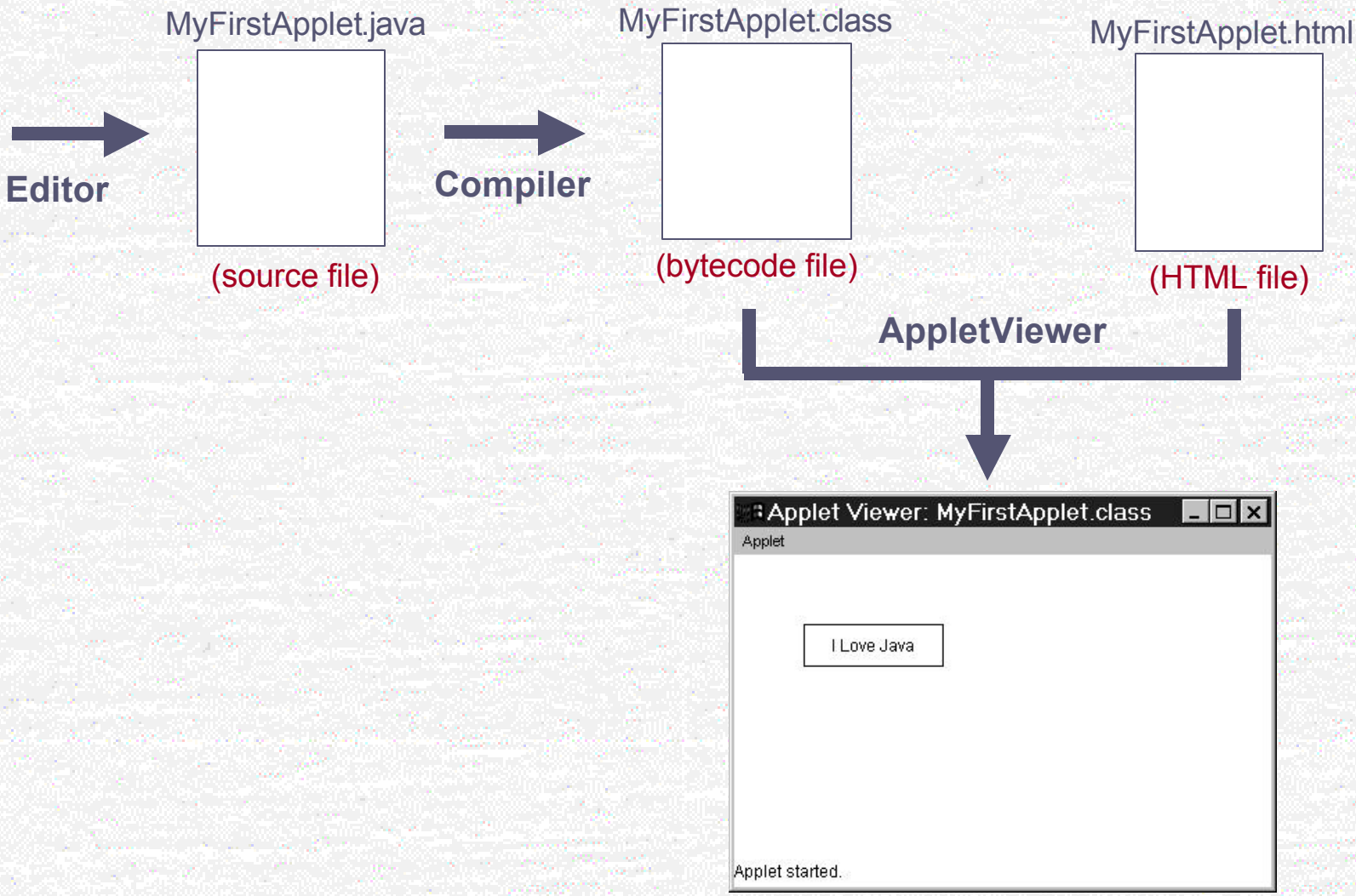


Executing Java Applets

- Basic steps for Java applications apply for applets as well.
- The main difference is that you need to define an HTML file. A Web browser or the AppletViewer needs this HTML file to execute an applet.
- An HTML file for the sample applet looks like this:

```
<HTML>
<BODY>
<APPLET CODE="MyFirstApplet.class" WIDTH=300 HEIGHT=190>
</APPLET>
</BODY>
</HTML>
```

Edit-Compile-Run Cycle for Applets




The End