21世纪高等院校计算机系列教材

影印版

Java程序设计

Java For Students (Fourth edition)

[英]Douglas Bell Mike Parr 著

中国水利水电出版社
www.waterpub.com.cn

Java
for Students

DOUGLAS BELL
MIKE PARR

Fourth edition

中国水利水电出版社
www.waterpub.com.cn

图书在版编目（CIP）数据

Java程序设计：第4版/（英）贝尔（Bell, D.），（英）帕尔（Parr, M.）著；—影印本，—北京：中国水利水电出版社，2006
（21世纪高等院校计算机系列教材）
书名原文：Java for Students: fourth edition
ISBN 7-5084-4109-5

Ⅰ.J… Ⅱ.①贝…②帕… Ⅲ.JAVA语言－程序设计—高等学校—教材—英文 Ⅳ.TP312

中国版本图书馆CIP数据核字（2006）第116322号

书　　名	Java程序设计（原书第4版）
作　　者	[英] Douglas Bell Mike Parr 著
出版发行	中国水利水电出版社（北京市三里河路6号 100044）
	网址：www.waterpub.com.cn
	E-mail：mchannel@263.net（万水）
	sales@waterpub.com.cn
	电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京市天竺颖华印刷厂
规　　格	787mm × 1092mm 16开本 29印张 638千字
版　　次	2006年10月第1版 2006年10月第1次印刷
印　　数	0001-3000册
定　　价	45.00元

凡购买我社图书，如有缺页、倒页、脱页的，本社营销中心负责调换

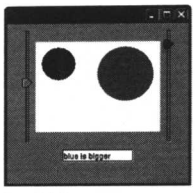


Figure 7.5 The bigger program.

(Figure 7.5). The program compares the sizes and reports on which one is the larger. Let us first write the program using the following i f statement:

```
if (redValue > blueValue) {
    textField.setText("red is bigger");
}
else {
    textField.setText("blue is bigger");
}
```

The library method f i l l O v a l is used to draw a filled circle whose diameter is equal to the value obtained from the corresponding slider. The complete program is:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class Bigger extends JFrame
                    implements ChangeListener {
    private JSlider redSlider;
    private JPanel panel;
    private JSlider blueSlider;
    private JTextField textField;
```



```
public static void main(String[] args) {
    Bigger demo = new Bigger();
    demo.setSize(300,300);
    demo.createGUI();
    demo.show();
}

private void createGUI() {
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    Container window = getContentPane();
    window.setLayout(new FlowLayout());

    redSlider = new JSlider(JSlider.VERTICAL);
    redSlider.addChangeListener(this);
    window.add(redSlider);

    panel = new JPanel();
    panel.setPreferredSize(new Dimension(200, 150));
    panel.setBackground(Color.white);
    window.add(panel);

    blueSlider = new JSlider(JSlider.VERTICAL);
    blueSlider.addChangeListener(this);
    window.add(blueSlider);

    textField = new JTextField(10);
    window.add(textField);
}
public void stateChanged(ChangeEvent e) {
    Graphics paper = panel.getGraphics();

    int redValue, blueValue;
    redValue = redSlider.getValue();
    blueValue = blueSlider.getValue();

    paper.setColor(Color.white);
    paper.fillRect(0, 0, 200, 150);
    paper.setColor(Color.red);
    paper.fillOval(10, 10, redValue, redValue);
    paper.setColor(Color.blue);
    paper.fillOval(100, 10, blueValue, blueValue);

    if (redValue > blueValue) {
        textField.setText("red is bigger");
    }
    else {
        textField.setText("blue is bigger");
    }
  }
}
```

Programming pitfalls highlight common programming mistakes and how to avoid them.

Example code is included in the text – this edition uses Swing throughout. Full code for all programs is available on the accompanying CD-ROM.

Programming pitfalls

- The method header must include type names. The following is wrong:

```
private void methodOne(x) {   // wrong
```

  Instead we must put, for example, the following:

```
private void methodOne(int x) {
```

- A method call must not include type names. For example, rather than:

```
methodOne(int y);   // wrong
```

  we put:

```
methodOne(y);
```

- When calling a method, we must supply the correct number of parameters and the correct types of parameters.
- We must arrange to consume a returned value in some way. The following style of call does not consume a return value:

```
someMethod(a, f);   // wrong
```

Grammar spot

- The general pattern for methods takes two forms. Firstly, for a method that does not return a result, we declare the method by:

```
private void methodName(format parameter list) {
    ... body
}
```

  and we call the method by a statement, as in:

```
methodName(actual parameter list);
```

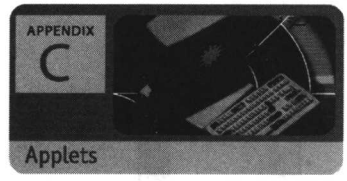
- For a method which returns a result, the form is:

```
private type methodName(actual parameter list) {
    ... body
}
```

  Any type or class can be specified as the returned type. We call the method as part of an expression. For example:

```
n = methodName(a, b);
```

Grammar spot identifies the correct way to write code, reinforcing the student's understanding of Java syntax.



Introduction

This book has been about writing 'applications'. They run under the control of your operating system and the Java code and corresponding class files are stored on your computer. Applets are different. The term means a small program. Compiled applet class files are uploaded to a web server computer, in the same folder as you might store your web pages. It is possible to specify that a web page links up to an applet. When a user downloads such a web page, the Java class code comes with it, and the applet runs in an area of the web browser window.

An applet example

Here we will look at the process of creating and running an applet. We will use the SumTextFields program of Chapter 6. Note that in Appendix B, we provided an AWT version of this program, and this is the version we will convert to an applet. The reason for this choice is because AWT applets will work with all browsers, but Swing support in browsers is not as widespread. Figure C.1 shows the applet running within a web browser. Here is the code:

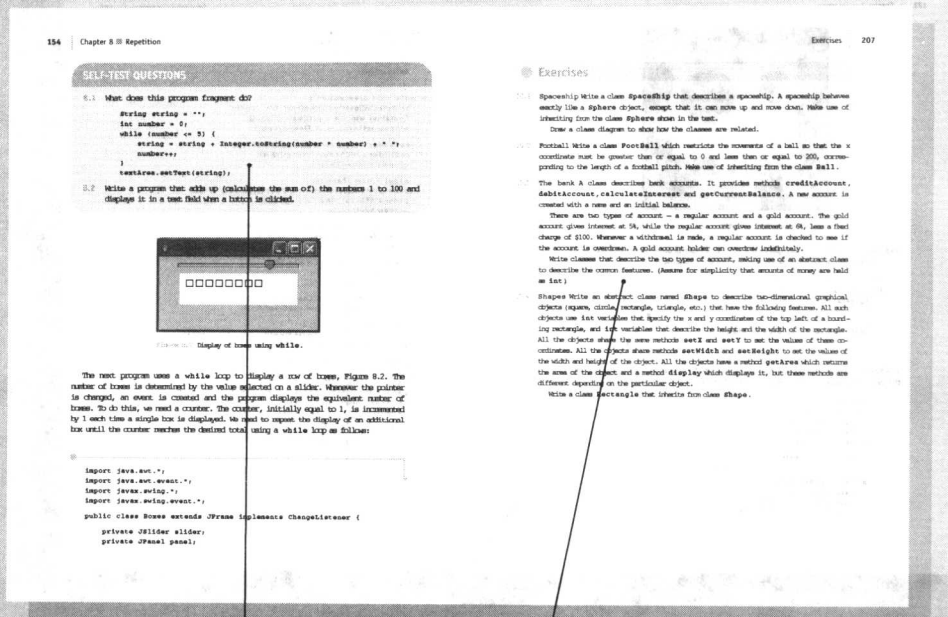
```
import java.awt.*;
import java.applet.Applet;
import java.awt.event.*;

public class SumTextFieldsApplet
    extends Applet implements ActionListener{

    private TextField number1Field, number2Field, sumField;
    private Label equalsLabel;
    private Button plusButton;
```


Appendices broaden the student's understanding of Java programming.

A guided tour



154 Chapter 8 ■ Repetition

SELF-TEST QUESTIONS

8.1 What does this program fragment do?

```
String string = "";
int number = 0;
while (number <= 5) {
    string = string + Integer.toString(number * number) + " ";
    number++;
}
textArea.setText(string);
```

8.2 Write a program that adds up (calculates the sum of) the numbers 1 to 100 and displays it in a text field when a button is clicked.

Figure 8.2 Display of boxes using while.

The next program uses a while loop to display a row of boxes, Figure 8.2. The number of boxes is determined by the value selected on a slider. Whenever the pointer is changed, an event is created and the program displays the equivalent number of boxes. To do this, we need a counter. The counter, initially equal to 1, is incremented by 1 each time a single box is displayed. We need to repeat the display of an additional box until the counter reaches the desired total, using a while loop as follows:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;

public class Boxes extends JFrame implements ChangeListener {

    private JSlider slider;
    private JPanel panel;
```

Exercises 207

Exercises

Spaceship Write a class **Spaceship** that describes a spaceship. A spaceship behaves exactly like a **Sphere** object, except that it can move up and move down. Make use of inheriting from the class **Sphere** shown in the text.

Draw a class diagram to show how the classes are related.

Football Write a class **FootBall** which restricts the movements of a ball so that the x coordinate must be greater than or equal to 0 and less than or equal to 200, corresponding to the length of a football pitch. Make use of inheriting from the class **Ball**.

The bank A class describes bank accounts. It provides methods **creditAccount**, **debitAccount**, **calculateInterest** and **getCurrentBalance**. A new account is created with a name and an initial balance.

There are two types of account – a regular account and a gold account. The gold account gives interest at 5%, while the regular account gives interest at 4%, less a fixed charge of £100. Whenever a withdrawal is made, a regular account is checked to see if the account is overdrawn. A gold account holder can overdraw indefinitely.

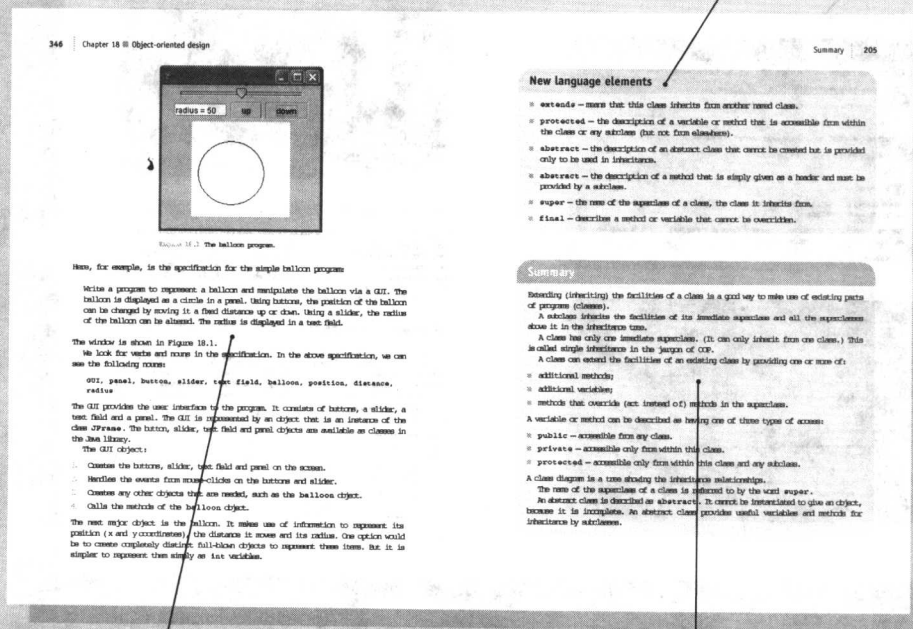
Write classes that describe the two types of account, making use of an abstract class to describe the common features. (Assume for simplicity that amounts of money are held as **int**.)

Shapes Write an abstract class named **Shape** to describe two-dimensional graphical objects (square, circle, rectangle, triangle, etc.) that have the following features. All such objects use **int** variables that specify the x and y coordinates of the top left of a bounding rectangle, and **int** variables that describe the height and the width of the rectangle. All the objects share the same methods **setX** and **setY** to set the values of these coordinates. All the objects share methods **setWidth** and **setHeight** to set the values of the width and height of the object. All the objects have a method **getArea** which returns the area of the object and a method **display** which displays it, but these methods are different depending on the particular object.

Write a class **Rectangle** that inherits from class **Shape**.

Numerous **self-test questions** throughout the book, and **exercises** at the end of every chapter allow the student to practice with the concepts until they fully understand them. The answers to the self-test questions appear at the end of each chapter.

New language elements reiterate the new syntax features introduced by the chapter.



346 Chapter 18 ■ Object-oriented design

Figure 18.1 The balloon program.

Here, for example, is the specification for the simple balloon program:

> Write a program to represent a balloon and manipulate the balloon via a GUI. The balloon is displayed as a circle in a panel. Using buttons, the position of the balloon can be changed by moving it a fixed distance up or down. Using a slider, the radius of the balloon can be altered. The radius is displayed in a text field.

The window is shown in Figure 18.1.

We look for verbs and nouns in the specification. In the above specification, we can see the following nouns:

```
GUI, panel, button, slider, text field, balloon, position, distance,
radius
```

The GUI provides the user interface to the program. It consists of buttons, a slider, a text field and a panel. The GUI is represented by an object that is an instance of the class **JFrame**. The button, slider, text field and panel objects are available as classes in the Java library.

The GUI object:

1. Creates the buttons, slider, text field and panel on the screen.
2. Handles the events from mouse-clicks on the buttons and slider.
3. Creates any other objects that are needed, such as the **balloon** object.
4. Calls the methods of the **balloon** object.

The next major object is the balloon. It makes use of information to represent its position (x and y coordinates), the distance it moves and its radius. One option would be to create completely distinct full-blown objects to represent these items. But it is simpler to represent them simply as **int** variables.

Summary 205

New language elements

- **extends** – means that this class inherits from another named class.
- **protected** – the description of a variable or method that is accessible from within the class or any subclass (but not from elsewhere).
- **abstract** – the description of an abstract class that cannot be created but is provided only to be used in inheritance.
- **abstract** – the description of a method that is simply given as a header and must be provided by a subclass.
- **super** – the name of the superclass of a class, the class it inherits from.
- **final** – describes a method or variable that cannot be overridden.

Summary

Extending (inheriting) the facilities of a class is a good way to make use of existing parts of programs (classes).

A subclass inherits the facilities of its immediate superclass and all the superclasses above it in the inheritance tree.

A class has only one immediate superclass. (It can only inherit from one class.) This is called single inheritance in the jargon of OOP.

A class can extend the facilities of an existing class by providing one or more of:

- additional methods;
- additional variables;
- methods that override (act instead of) methods in the superclass.

A variable or method can be described as having one of three types of access:

- **public** – accessible from any class.
- **private** – accessible only from within this class.
- **protected** – accessible only from within this class and any subclass.

A class diagram is a tree showing the inheritance relationships.

The name of the superclass of a class is referred to by the word **super**.

An abstract class is described as **abstract**. It cannot be instantiated to give an object, because it is incomplete. An abstract class provides useful variables and methods for inheritance by subclasses.

Programs that use graphical images (particularly GUIs) rather than text input-output programs are used throughout. This demonstrates the creative and exciting side of programming which helps the student learn concepts faster.

Summaries offer a concise round-up of the key concepts covered by each chapter. They tie in with the objectives listed at the beginning of the chapter and are a great reference and revision aid.

Introduction

What this book will tell you

This book explains how to write Java programs that run either as independent applications or as applets (part of a web page).

This book is for novices

If you have never done any programming before – if you are a complete novice – this book is for you. This book assumes no prior knowledge of programming. It starts from scratch. It is written in a simple, direct style for maximum clarity. It is aimed primarily at first-year undergraduates at universities and colleges, but it is also suitable for novices studying alone.

Why Java?

Java is probably one of the best programming languages to learn and use because of the following features.

Java is small and beautiful

The designers of Java have deliberately left out all the superfluous features of programming languages; they cut the design to the bone. The result is a language that has all the necessary fetures, combined in an elegant and logical way. The design is lean and mean. It is easy to learn, but powerful.

Java is object oriented

Object-oriented languages are the latest and most successful approach to programming. Object-oriented programming is the most popular approach to programming. Java is completely object oriented from the ground up. It is not a language that has had object-orientedness grafted onto it as an afterthought.

Java supports the Internet

A major motivation for Java is to enable people to develop programs that use the Internet and the World-Wide Web. Java applets can easily be invoked from web browsers such as Internet Explorer to provide valuable and spectacular facilities. In addition, Java programs can be easily transmitted around the Internet and run on any computer.

Java is general purpose

Java is a truly general-purpose language. Anything that C++, Visual Basic, etc., can do, so can Java.

Java is platform independent

Java programs will run on almost all computers and with nearly all operating systems – unchanged! Try that with any other programming language. (You almost certainly can't!) This is summed up in the slogan 'write once – run anywhere'.

Java is robust

The Java compiler carries out many stringent checks as it prepares a program for execution. Once a program has been corrected and compiles without errors, it often performs correctly. However, if a Java program goes wrong (and programs do have that tendency), it won't create mayhem, damage and uncertainty.

Java has libraries

Because Java is a small language, most of its functionality is provided by pieces of program held in libraries. A whole host of library software is available to do graphics, access the Internet, provide graphical user interfaces (GUIs) and many other things.

Exercises are good for you

If you were to read this book time and again until you could recite it backwards, you still wouldn't be able to write programs. The practical work of writing programs and program fragments is vital to becoming fluent and confident at programming.

There are exercises for the reader at the end of each chapter. Please do some of them to enhance your ability to program.

There are also short self-test questions throughout the text with answers at the end of the chapter, so that you can check you have understood things properly.

What's included?

This book explains the fundamentals of programming:

- variables;

- assignment;
- input and output;
- calculation;
- graphics and windows programming;
- selection using `if`;
- repetition using `while`.

It also covers integer numbers, floating-point numbers and character strings. Arrays are also described. All these are topics that are fundamental, whatever kind of programming you go on to do.

This book also thoroughly addresses the object-oriented aspects of programming:

- using library classes;
- writing classes;
- using objects;
- using methods.

We also look at some of the more sophisticated aspects of object-oriented programming, like:

- inheritance;
- polymorphism;
- interfaces.

What's not included

This book describes the essentials of Java. It does not explain the bits and pieces, the bells and whistles. Thus the reader is freed from unnecessary detail and can concentrate on mastering Java and programming in general.

Applications or applets?

There are two distinct types of Java program:

- a distinct free-standing program (this is called an application);
- a program invoked from a web browser (this is called an applet).

In this book we concentrate on applications, because we believe that this is the main way in which Java is being used. (We explain how to run applets in an appendix.)

Graphics or text?

Throughout the text we have emphasized programs that use graphical images rather than text input and output. We think they are more fun, more interesting and clearly demonstrate all the important principles of programming. We haven't ignored programs that input and output text – they are included, but they come second best.

Graphical user interfaces (GUIs)

The programs we present use many of the features of a GUI, such as windows, buttons, scrollbars and using the mouse in lots of different ways.

AWT or Swing?

There are two Java mechanisms for creating and using GUIs – AWT and Swing. The Swing set of user-interface components is more complete and powerful than the AWT set. This book uses the Swing approach because it is being used more widely.

The sequence of material

Programming involves many challenging ideas, and one of the problems of writing a book about programming is deciding how and when to introduce new ideas. We introduce simple ideas early and more sophisticated ideas later on. We use objects from an early stage. Then later we see how to write new objects. Our approach is to start with ideas like variables and assignment, then introduce selection and looping, and then go on to objects and classes (the object-oriented features). We also wanted to make sure that the fun element of programming is paramount, so we use graphics right from the start.

Bit by bit

In this book we introduce new ideas carefully one at a time, rather than all at once. So there is a single chapter on writing methods, for example.

Computer applications

Computers are used in many different applications and this book uses examples from all these areas:

- information processing;
- games;
- scientific calculations.

The reader can choose to concentrate on those application areas of interest and spend less time on the other areas.

Different kinds of programming

There are many different kinds of programming – examples are procedural, logic, functional, spreadsheet, visual and object-oriented programming. This book is about the dominant type of programming – object-oriented programming (OOP) – as practised in languages like Visual Basic, C++, C#, Eiffel and Smalltalk.

Which version of Java?

Java is evolving, but slowly. From time to time Sun releases a new version of the Java Software Development Kit (SDK). A new version usually means additional items in the libraries – and a new version number. All versions from 1.2 have the generic name of Java 2. You can use this book with any version from version 1.2 onwards.

Have fun

Programming is creative and interesting, particularly in Java. Please have fun!

Visit our web site

All the programs presented in this book are available on our web site, which can be reached via: **www.booksites.net/bell.**

Changes to this edition

If you have used earlier editions of this book, you might like to know what is different about this edition.

In making changes we have tried to keep to the spirit of the original and at the same time simplify things where possible. We have also tried to follow the main trends in using Java. So while the older editions used applets and the AWT for GUIs, this edition uses applications and Swing. The older editions used the method `paint` extensively and everyone found it difficult to understand. So we have eliminated `paint` to give major simplifications.

We have also made minor changes, such as using the term 'call' rather than the longwinded term 'invoke'. We have also used `double` variables instead of `float` because declaring literals is easier.

There used to be a chapter on applet architecture, but we were never sure that it worked. With the elimination of method `paint` and the simpler structure of applications, we don't think this chapter is needed anymore. There was also a chapter on GUI components, which seemed out of place. So this chapter has been removed and incorporated within an improved appendix on the Java library, which presents sample programs.

The topic of abstract classes has been moved into the chapter on inheritance, where it rightly belongs. We have created a new chapter on array lists, which serves as an introduction to data structures.

We hope you like the changes.

Any comments on this book?

If you want to email the authors, we are at D.H.Bell@shu.ac.uk and M.Parr@shu.ac.uk. We look forward to hearing from you.

Contents

Detailed contents