Build graphical applications that use buttons.

- Use a `JPanel` container to group multiple user-interface components together.
- You specify button click actions through classes that implement the `ActionListener` interface.

Classes, Objects, and Methods Introduced in this Chapter

```
java.awt.Component
   addMouseListener
   repaint
   setPreferredSize
java.awt.Container
   add
java.awt.Dimension
java.awt.Rectangle
   setLocation
java.awt.event.ActionListener
   actionPerformed
java.awt.event.MouseEvent
   getX
   getY
java.awt.event.MouseListener
   mouseClicked
   mouseEntered
   mouseExited
   mousePressed
   mouseReleased
javax.swing.AbstractButton
   addActionListener
javax.swing.JButton
javax.swing.JLabel
javax.swing.JPanel
javax.swing.Timer
   start
   stop
```

Media Resources

www.wiley.com/go/global/horstmann

- ***Worked Example*** Investigating Number Sequences
- Lab Exercises
- ***Animation*** Polymorphism
- Practice Quiz
- Code Completion Exercises

Review Exercises

★ **R8.1** Suppose `C` is a class that implements the interfaces `I` and `J`. Which of the following assignments require a cast?

```
C c = . . .;
I i = . . .;
J j = . . .;
```

a. `c = i;`

b. `j = c;`

c. `i = j;`

★ **R8.2** Suppose `C` is a class that implements the interfaces `I` and `J`, and suppose `i` is declared as

```
I i = new C();
```

Which of the following statements will throw an exception?

a. `C c = (C) i;`

b. `J j = (J) i;`

c. `i = (I) null;`

★ **R8.3** Suppose the class `Sandwich` implements the `Edible` interface, and you are given the variable declarations

```
Sandwich sub = new Sandwich();
Rectangle cerealBox = new Rectangle(5, 10, 20, 30);
Edible e = null;
```

Which of the following assignment statements are legal?

a. `e = sub;`

b. `sub = e;`

c. `sub = (Sandwich) e;`

d. `sub = (Sandwich) cerealBox;`

e. `e = cerealBox;`

f. `e = (Edible) cerealBox;`

g. `e = (Rectangle) cerealBox;`

h. `e = (Rectangle) null;`

★★ **R8.4** How does a cast such as `(BankAccount) x` differ from a cast of number values such as `(int) x`?

★★ **R8.5** The classes `Rectangle2D.Double`, `Ellipse2D.Double`, and `Line2D.Double` implement the `Shape` interface. The `Graphics2D` class depends on the `Shape` interface but not on the rectangle, ellipse, and line classes. Draw a UML diagram denoting these facts.

★★ **R8.6** Suppose `r` contains a reference to a `new Rectangle(5, 10, 20, 30)`. Which of the following assignments is legal? (Look inside the API documentation to check which interfaces the `Rectangle` class implements.)

a. `Rectangle a = r;`

b. `Shape b = r;`

c. `String c = r;`

d. `ActionListener d = r;`

e. `Measurable e = r;`

f. `Serializable f = r;`

g. `Object g = r;`

★★ **R8.7** Classes such as `Rectangle2D.Double`, `Ellipse2D.Double` and `Line2D.Double` implement the `Shape` interface. The `Shape` interface has a method

```
Rectangle getBounds()
```

that returns a rectangle completely enclosing the shape. Consider the method call:

```
Shape s = . . .;
Rectangle r = s.getBounds();
```

Explain why this is an example of polymorphism.

★★★ **R8.8** In Java, a method call such as `x.f()` uses dynamic method lookup—the exact method to be called depends on the type of the object to which `x` refers. Give two kinds of method calls that do not use dynamic method lookup in Java.

★★ **R8.9** Suppose you need to process an array of employees to find the average and the highest salaries. Discuss what you need to do to use the implementation of the `DataSet` class in Section 8.1 (which processes `Measurable` objects). What do you need to do to use the second implementation (in Section 8.4)? Which is easier?

★★★ **R8.10** What happens if you add a `String` object to the implementation of the `DataSet` class in Section 8.1? What happens if you add a `String` object to a `DataSet` object of the implementation in Section 8.4 that uses a `RectangleMeasurer` class?

★ **R8.11** How would you reorganize the `DataSetTester3` program if you needed to make `RectangleMeasurer` into a top-level class (that is, not an inner class)?

★★ **R8.12** What is a callback? Can you think of another use for a callback for the `DataSet` class? (*Hint:* Exercise P8.12.)

★★ **R8.13** Consider this top-level and inner class. Which variables can the `f` method access?

```
public class T
{
   private int t;

   public void m(final int x, int y)
   {
      int a;
      final int b;

      class C implements I
      {
         public void f()
         {
            . . .
         }
      }

      final int c;
      . . .
   }
}
```

★★ **R8.14** What happens when an inner class tries to access a non-`final` local variable? Try it out and explain your findings.

★★★G **R8.15** How would you reorganize the `InvestmentViewer1` program if you needed to make `AddInterestListener` into a top-level class (that is, not an inner class)?

★G **R8.16** What is an event object? An event source? An event listener?

★G **R8.17** From a programmer's perspective, what is the most important difference between the user interfaces of a console application and a graphical application?

★G **R8.18** What is the difference between an `ActionEvent` and a `MouseEvent`?

★★G **R8.19** Why does the `ActionListener` interface have only one method, whereas the `MouseListener` has five methods?

★★G **R8.20** Can a class be an event source for multiple event types? If so, give an example.

★★G **R8.21** What information does an action event object carry? What additional information does a mouse event object carry?

★★★G **R8.22** Why are we using inner classes for event listeners? If Java did not have inner classes, could we still implement event listeners? How?

★★G **R8.23** What is the difference between the `paintComponent` and `repaint` methods?

★G **R8.24** What is the difference between a frame and a panel?

Programming Exercises

★ **P8.1** Have the `Die` class of Chapter 5 implement the `Measurable` interface. Generate dice, cast them, and add them to the implementation of the `DataSet` class in Section 8.1. Display the average.

★ **P8.2** Implement a class `Quiz` that implements the `Measurable` interface. A quiz has a score and a letter grade (such as `B+`). Use the implementation of the `DataSet` class in Section 8.1 to process a collection of quizzes. Display the average score and the quiz with the highest score (both letter grade and score).

★ **P8.3** A person has a name and a height in centimeters. Use the implementation of the `DataSet` class in Section 8.4 to process a collection of `Person` objects. Display the average height and the name of the tallest person.

★ **P8.4** Modify the implementation of the `DataSet` class in Section 8.1 (the one processing `Measurable` objects) to also compute the minimum data element.

★ **P8.5** Modify the implementation of the `DataSet` class in Section 8.4 (the one using a `Measurer` object) to also compute the minimum data element.

★ **P8.6** Using a different `Measurer` object, process a set of `Rectangle` objects to find the rectangle with the largest perimeter.

★★★ **P8.7** Enhance the `DataSet` class so that it can either be used with a `Measurer` object or for processing `Measurable` objects. *Hint:* Supply a constructor with no parameters that implements a `Measurer` that processes `Measurable` objects.

★ **P8.8** Modify the `display` method of the `LastDigitDistribution` class of Worked Example 8.1 so that it produces a histogram, like this:

```
0: *************
1: ******************
2: *************
```

Scale the bars so that widest one has length 40.

★★ **P8.9** Write a class `PrimeSequence` that implements the `Sequence` interface of Worked Example 8.1, producing the sequence of prime numbers.

★ **P8.10** Add a method `hasNext` to the `Sequence` interface of Worked Example 8.1 that returns `false` if the sequence has no more values. Implement a class `MySequence` producing a sequence of real data of your choice, such as populations of cities or countries, temperatures, or stock prices. Obtain the data from the Internet and reformat the values so that they are placed into an array. Return one value at a time in the `next` method, until you reach the end of the data. Your `SequenceTester` class should display all data in the sequence and check whether the last digits are randomly distributed.

★ **P8.11** Provide a class `FirstDigitDistribution` that works just like the `LastDigitDistribution` class of Worked Example 8.1, except that it counts the distribution of the first digit of each value. (It is a well-known fact that the first digits of random values are *not* uniformly distributed. This fact has been used to detect accounting fraud, when sequences of transaction amounts had an unnatural distribution of their first digits.)

★ **P8.12** Declare an interface `Filter` as follows:

```
public interface Filter
{
   boolean accept(Object x);
}
```

Modify the implementation of the `DataSet` class in Section 8.4 to use both a `Measurer` and a `Filter` object. Only objects that the filter accepts should be processed. Demonstrate your modification by having a data set process a collection of bank accounts, filtering out all accounts with balances less than $1,000.

★ **P8.13** The standard Java library provides a `Comparable` interface:

```
public interface Comparable
{
   /**
      Compares this object with another.
      @param other the object to be compared
      @return a negative integer, zero, or a positive integer if this object
      is less than, equal to, or greater than, other
   */
   public int compareTo(Object other);
}
```

Modify the `DataSet` class of Section 8.1 to accept `Comparable` objects. With this interface, it is no longer meaningful to compute the average. The `DataSet` class should record the minimum and maximum data values. Test your modified `DataSet` class by adding a number of `String` objects. (The `String` class implements the `Comparable` interface.)

★ **P8.14** Modify the `Coin` class to have it implement the `Comparable` interface described in Exercise P8.13.

★ **P8.15** The `System.out.printf` method has predefined formats for printing integers, floating-point numbers, and other data types. But it is also extensible. If you use the `S` format, you can print any class that implements the `Formattable` interface. That interface has a single method:

```
void formatTo(Formatter formatter, int flags, int width, int precision)
```

In this exercise, you should make the `BankAccount` class implement the `Formattable` interface. Ignore the flags and precision and simply format the bank balance, using the given width. In order to achieve this task, you need to get an `Appendable` reference like this:

```
Appendable a = formatter.out();
```

`Appendable` is another interface with a method

```
void append(CharSequence sequence)
```

`CharSequence` is yet another interface that is implemented by (among others) the `String` class. Construct a string by first converting the bank balance into a string and then padding it with spaces so that it has the desired width. Pass that string to the `append` method.

★ **P8.16** Enhance the `formatTo` method of Exercise P8.15 by taking into account the precision.

★T **P8.17** Consider the task of writing a program that plays TicTacToe against a human opponent. A user interface `TicTacToeUI` reads the user's moves and displays the computer's moves and the board. A class `TicTacToeStrategy` determines the next move that the computer makes. A class `TicTacToeBoard` represents the current state of the board. Complete all classes except for the strategy class. Instead, use a mock class that simply picks the first available empty square.

★★T **P8.18** Consider the task of translating a plain text book from Project Gutenberg (`http://gutenberg.org`) to HTML. For example, here is the start of the first chapter of Tolstoy's Anna Karenina:

```
Chapter 1

Happy families are all alike; every unhappy family is unhappy in
its own way.

Everything was in confusion in the Oblonskys' house. The wife
had discovered that the husband was carrying on an intrigue with
a French girl, who had been a governess in their family, and she
had announced to her husband that she could not go on living in
the same house with him ...
```

The equivalent HTML is:

```
<h1>Chapter 1</h1>
<p>Happy families are all alike; every unhappy family is unhappy in
its own way.</p>
<p>Everything was in confusion in the Oblonskys&rsquo; house. The wife
had discovered that the husband was carrying on an intrigue with
a French girl, who had been a governess in their family, and she
had announced to her husband that she could not go on living in
the same house with him ...</p>
```

The HTML conversion can be carried out in two steps. First, the plain text is assembled into *segments*, blocks of text of the same kind (heading, paragraph, and so on). Then each segment is converted, by surrounding it with the HTML tags and converting special characters.

Plain Text	HTML
“ ”	`&ldquo;` (left) *or* `&rdquo;` (right)
‘ ’	`&lsquo;` (left) *or* `&rsquo;` (right)
—	`&emdash;`
<	`<`
>	`>`
&	`&`

Fetching the text from the Internet and breaking it into segments is a challenging task. Provide an interface and a mock implementation. Combine it with a class that uses the mock implementation to finish the formatting task.

★★★G **P8.19** Write a method `randomShape` that randomly generates objects implementing the `Shape` interface: some mixture of rectangles, ellipses, and lines, with random positions. Call it 10 times and draw all of them.

P8.20 Enhance the `ButtonViewer` program so that it prints a message "I was clicked *n* times!" whenever the button is clicked. The value *n* should be incremented with each click.

P8.21 Enhance the `ButtonViewer` program so that it has two buttons, each of which prints a message "I was clicked *n* times!" whenever the button is clicked. Each button should have a separate click count.

P8.22 Enhance the `ButtonViewer` program so that it has two buttons labeled A and B, each of which prints a message "Button *x* was clicked!", where *x* is A or B.

P8.23 Implement a `ButtonViewer` program as in Exercise P8.22, using only a single listener class.

P8.24 Enhance the `ButtonViewer` program so that it prints the time at which the button was clicked.

P8.25 Implement the `AddInterestListener` in the `InvestmentViewer1` program as a regular class (that is, not an inner class). *Hint:* Store a reference to the bank account. Add a constructor to the listener class that sets the reference.

P8.26 Implement the `AddInterestListener` in the `InvestmentViewer2` program as a regular class (that is, not an inner class). *Hint:* Store references to the bank account and the label in the listener. Add a constructor to the listener class that sets the references.

P8.27 Write a program that demonstrates the growth of a roach population. Start with two roaches and double the number of roaches with each button click.

P8.28 Write a program that uses a timer to print the current time once a second. *Hint:* The following code prints the current time:

```
Date now = new Date();
System.out.println(now);
```

The `Date` class is in the `java.util` package.

P8.29 Change the `RectangleComponent` for the animation program in Section 8.10 so that the rectangle bounces off the edges of the component rather than simply moving outside. (See `ch08/timer/` in your source code.)

G P8.30 Write a program that animates a car so that it moves across a frame.

G P8.31 Write a program that animates two cars moving across a frame in opposite directions (but at different heights so that they don't collide.)

G P8.32 Change the `RectangleComponent` for the mouse listener program in Section 8.11 so that a new rectangle is added to the component whenever the mouse is clicked. *Hint:* Keep an `ArrayList<Rectangle>` and draw all rectangles in the `paintComponent` method. (See `ch08/mouse/` in your source code.)

★G P8.33 Write a program that prompts the user to enter the *x*- and *y*-positions of the center and a radius, using `JOptionPane` dialogs.When the user clicks a "Draw" button, draw a circle with that center and radius in a component.

★G P8.34 Write a program that allows the user to specify a circle by typing the radius in a `JOptionPane` and then clicking on the center. Note that you don't need a "Draw" button.

★G P8.35 Write a program that allows the user to specify a circle with two mouse presses, the first one on the center and the second on a point on the periphery. *Hint:* In the mouse press handler, you must keep track of whether you already received the center point in a previous mouse press.

Programming Projects

Project 8.1 Design an interface `MoveableShape` that can be used as a generic mechanism for animating a shape. A moveable shape must have two methods: `move` and `draw`. Write a generic `AnimationPanel` that paints and moves any `MoveableShape` (or array list of `MoveableShape` objects if you covered Chapter 6). Supply moveable rectangle and car shapes.

Project 8.2 Your task is to design a general program for managing board games with two players. Your program should be flexible enough to handle games such as tic-tac-toe, chess, or the Game of Nim of Project 5.2.

Design an interface `Game` that describes a board game. Think about what your program needs to do. It asks the first player to input a move—a string in a game-specific format, such as `Be3` in chess. Your program knows nothing about specific games, so the `Game` interface must have a method such as

```
boolean isValidMove(String move)
```

Once the move is found to be valid, it needs to be executed—the interface needs another method `executeMove`. Next, your program needs to check whether the game is over. If not, the other player's move is processed. You should also provide some mechanism for displaying the current state of the board.

Design the `Game` interface and provide two implementations of your choice—such as `Nim` and `Chess` (or `TicTacToe` if you are less ambitious). Your `GamePlayer` class should manage a `Game` reference without knowing which game is played, and process the moves from both players. Supply two programs that differ only in the initialization of the `Game` reference.

Answers to Self-Check Questions

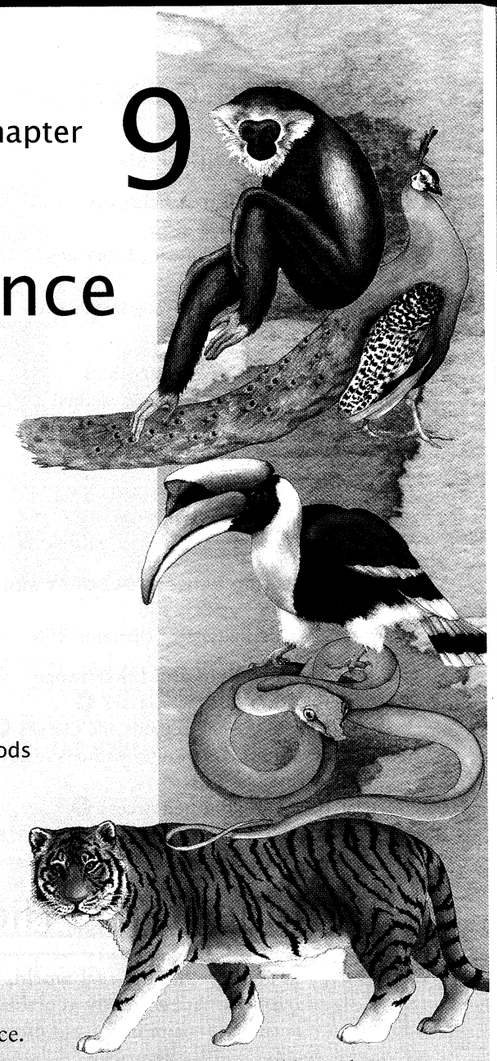
1. It must implement the `Measurable` interface, and its `getMeasure` method must return the population.
2. The `Object` class doesn't have a `getMeasure` method, and the `add` method invokes the `getMeasure` method.
3. Only if `x` actually refers to a `BankAccount` object.
4. No—a `Coin` reference can be converted to a `Measurable` reference, but if you attempt to cast that reference to a `BankAccount`, an exception occurs.
5. `Measurable` is an interface. Interfaces have no instance variables and no method implementations.
6. That variable never refers to a `Measurable` object. It refers to an object of some class—a class that implements the `Measurable` interface.
7. The code fragment prints 500.05. Each call to `add` results in a call `x.getMeasure()`. In the first call, `x` is a `BankAccount`. In the second call, `x` is a `Coin`. A different `getMeasure`

method is called in each case. The first call returns the account balance, the second one the coin value.

8. The `String` class doesn't implement the `Measurable` interface.
9. Implement a class `StringMeasurer` that implements the `Measurer` interface.
10. A measurer measures an object, whereas `getMeasure` measures "itself", that is, the implicit parameter.
11. Inner classes are convenient for insignificant classes. Also, their methods can access local and instance variables from the surrounding scope.
12. Four: one for the outer class, one for the inner class, and two for the `DataSet` and `Measurer` classes.
13. You want to implement the `GradingProgram` class in terms of that interface so that it doesn't have to change when you switch between the mock class and the actual class.
14. Because the developer of `GradingProgram` doesn't have to wait for the `GradeBook` class to be complete.
15. The `button` object is the event source. The `listener` object is the event listener.
16. The `ClickListener` class implements the `ActionListener` interface.
17. Direct access is simpler than the alternative—passing the variable as a parameter to a constructor or method.
18. The local variable must be declared as `final`.
19. First add `label` to the `panel`, then add `button`.
20. The `actionPerformed` method does not access that variable.
21. The timer needs to call some method whenever the time interval expires. It calls the `actionPerformed` method of the listener object.

Chapter 9

Inheritance



CHAPTER GOALS

- To learn about inheritance
- To understand how to inherit and override superclass methods
- To be able to invoke superclass constructors
- To learn about protected and package access control
- To understand the common superclass `Object` and how to override its `toString` and `equals` methods
- To use inheritance for customizing user interfaces

In this chapter, we discuss the important concept of inheritance. Specialized classes can be created that inherit behavior from more general classes. You will learn how to implement inheritance in Java, and how to make use of the `Object` class—the most general class in the inheritance hierarchy.

Resources

- ***Worked Example*** Implementing an Employee Hierarchy for Payroll Processing
- Lab Exercises
- ***Animation*** Inheritance
- Practice Quiz
- Code Completion Exercises

Exercises

R9.1 What is the balance of `b` after the following operations?

```
SavingsAccount b = new SavingsAccount(10);
b.deposit(5000);
b.withdraw(b.getBalance() / 2);
b.addInterest();
```

R9.2 Describe all constructors of the `SavingsAccount` class. List all methods that are inherited from the `BankAccount` class. List all methods that are added to the `SavingsAccount` class.

R9.3 Can you convert a superclass reference into a subclass reference? A subclass reference into a superclass reference? If so, give examples. If not, explain why not.

R9.4 Identify the superclass and the subclass in each of the following pairs of classes.

a. `Employee, Manager`
b. `Polygon, Triangle`
c. `GraduateStudent, Student`
d. `Person, Student`
e. `Employee, GraduateStudent`
f. `BankAccount, CheckingAccount`
g. `Vehicle, Car`
h. `Vehicle, Minivan`
i. `Car, Minivan`
j. `Truck, Vehicle`

R9.5 Suppose the class `Sub` extends the class `Sandwich`. Which of the following assignments are legal?

```
Sandwich x = new Sandwich();
Sub y = new Sub();
```

a. `x = y;`
b. `y = x;`
c. `y = new Sandwich();`
d. `x = new Sub();`

R9.6 Draw an inheritance diagram that shows the inheritance relationships between the classes:

- `Person`
- `Employee`
- `Student`
- `Instructor`
- `Classroom`
- `Object`

R9.7 In an object-oriented traffic simulation system, we have the following classes:

- `Vehicle`
- `Car`
- `Truck`
- `Sedan`
- `Coupe`
- `PickupTruck`
- `SportUtilityVehicle`
- `Minivan`
- `Bicycle`
- `Motorcycle`

Draw an inheritance diagram that shows the relationships between these classes.

R9.8 What inheritance relationships would you establish among the following classes?

- `Student`
- `Professor`
- `TeachingAssistant`
- `Employee`
- `Secretary`
- `DepartmentChair`
- `Janitor`
- `SeminarSpeaker`
- `Person`
- `Course`
- `Seminar`
- `Lecture`
- `ComputerLab`

R9.9 Which of these conditions returns `true`? Check the Java documentation for the inheritance patterns.

```
Rectangle r = new Rectangle(5, 10, 20, 30);
```

a. `if (r instanceof Rectangle) . . .`
b. `if (r instanceof Point) . . .`
c. `if (r instanceof Rectangle2D.Double) . . .`
d. `if (r instanceof RectangularShape) . . .`
e. `if (r instanceof Object) . . .`
f. `if (r instanceof Shape) . . .`

R9.10 Explain the two meanings of the `super` reserved word. Explain the two meanings of the `this` reserved word. How are they related?

R9.11 (Tricky.) Consider the two calls

```
public class D extends B
{
   public void f()
   {
      this.g(); // 1
   }
   public void g()
   {
      super.g(); // 2
   }
   . . .
}
```

Which of them is an example of polymorphism?

R9.12 Consider this program:

```
public class AccountPrinter
{
```

Media Resources

www.wiley.com/go/global/horstmann

- ***Worked Example*** Implementing an Employee Hierarchy for Payroll Processing
- Lab Exercises
- ***Animation*** Inheritance
- Practice Quiz
- Code Completion Exercises

Review Exercises

★ **R9.1** What is the balance of b after the following operations?

```
SavingsAccount b = new SavingsAccount(10);
b.deposit(5000);
b.withdraw(b.getBalance() / 2);
b.addInterest();
```

★ **R9.2** Describe all constructors of the SavingsAccount class. List all methods that are inherited from the BankAccount class. List all methods that are added to the SavingsAccount class.

★★ **R9.3** Can you convert a superclass reference into a subclass reference? A subclass reference into a superclass reference? If so, give examples. If not, explain why not.

★★ **R9.4** Identify the superclass and the subclass in each of the following pairs of classes.

a. Employee, Manager
b. Polygon, Triangle
c. GraduateStudent, Student
d. Person, Student
e. Employee, GraduateStudent
f. BankAccount, CheckingAccount
g. Vehicle, Car
h. Vehicle, Minivan
i. Car, Minivan
j. Truck, Vehicle

★ **R9.5** Suppose the class Sub extends the class Sandwich. Which of the following assignments are legal?

```
Sandwich x = new Sandwich();
Sub y = new Sub();
```

a. x = y;
b. y = x;
c. y = new Sandwich();
d. x = new Sub();

★ **R9.6** Draw an inheritance diagram that shows the inheritance relationships between the classes:

- Person
- Employee
- Student
- Instructor
- Classroom
- Object

★★ **R9.7** In an object-oriented traffic simulation system, we have the following classes:

- Vehicle
- Car
- Truck
- Sedan
- Coupe
- PickupTruck
- SportUtilityVehicle
- Minivan
- Bicycle
- Motorcycle

Draw an inheritance diagram that shows the relationships between these classes.

★★ **R9.8** What inheritance relationships would you establish among the following classes?

- Student
- Professor
- TeachingAssistant
- Employee
- Secretary
- DepartmentChair
- Janitor
- SeminarSpeaker
- Person
- Course
- Seminar
- Lecture
- ComputerLab

★★★ **R9.9** Which of these conditions returns true? Check the Java documentation for the inheritance patterns.

```
Rectangle r = new Rectangle(5, 10, 20, 30);
```

a. if (r instanceof Rectangle) . . .
b. if (r instanceof Point) . . .
c. if (r instanceof Rectangle2D.Double) . . .
d. if (r instanceof RectangularShape) . . .
e. if (r instanceof Object) . . .
f. if (r instanceof Shape) . . .

★★ **R9.10** Explain the two meanings of the super reserved word. Explain the two meanings of the this reserved word. How are they related?

★★★ **R9.11** (Tricky.) Consider the two calls

```
public class D extends B
{
   public void f()
   {
      this.g(); // 1
   }
   public void g()
   {
      super.g(); // 2
   }
   . . .
}
```

Which of them is an example of polymorphism?

★★★ **R9.12** Consider this program:

```
public class AccountPrinter
{
```

```
public static void main(String[] args)
{
   SavingsAccount momsSavings
         = new SavingsAccount(0.5);

   CheckingAccount harrysChecking
         = new CheckingAccount(0);

   . . .
   endOfMonth(momsSavings);
   endOfMonth(harrysChecking);
   printBalance(momsSavings);
   printBalance(harrysChecking);
}

public static void endOfMonth(SavingsAccount savings)
{
   savings.addInterest();
}

public static void endOfMonth(CheckingAccount checking)
{
   checking.deductFees();
}

public static void printBalance(BankAccount account)
{
   System.out.println("The balance is $"
         + account.getBalance());
}
```

Do the calls to the `endOfMonth` methods use dynamic method invocation? Inside the `printBalance` method, does the call to `getBalance` use dynamic method invocation?

★ **R9.13** Explain the terms *shallow copy* and *deep copy*.

★ **R9.14** What access attribute should instance variables have? What access attribute should static variables have? How about static final variables?

★ **R9.15** What access attribute should instance methods have? Does the same hold for static methods?

★★ **R9.16** The static variables `System.in` and `System.out` are public. Is it possible to overwrite them? If so, how?

★★ **R9.17** Why are public instance variables dangerous? Are public static variables more dangerous than public instance variables?

Programming Exercises

★ **P9.1** Enhance the `addInterest` method of the `SavingsAccount` class to compute the interest on the *minimum* balance since the last call to `addInterest`. *Hint:* You need to modify the `withdraw` method as well, and you need to add an instance variable to remember the minimum balance.

★★ **P9.2** Add a `TimeDepositAccount` class to the bank account hierarchy. The time deposit account is just like a savings account, but you promise to leave the money in the account for a particular number of months, and there is a $20 penalty for early withdrawal. Construct the account with the interest rate and the number of months to maturity. In the `addInterest` method, decrement the count of months. If the count is positive during a withdrawal, charge the withdrawal penalty.

★ **P9.3** Add a class `NumericQuestion` to the question hierarchy of How To 9.1. If the response and the expected answer differ by no more than 0.01, then accept it as correct.

★★ **P9.4** Add a class `FillInQuestion` to the question hierarchy of How To 9.1. An object of this class is constructed with a string that contains the answer, surrounded by _ _, for example, `"The inventor of Java was _James Gosling_"`. The question should be displayed as

```
The inventor of Java was ____
```

★ **P9.5** Modify the `checkAnswer` method of the `Question` class of How To 9.1 so that it does not take into account different spaces or upper/lowercase characters. For example, the response `" JAMES gosling"` should match an answer of `"James Gosling"`.

★ **P9.6** Add a class `MultiChoiceQuestion` to the question hierarchy of How To 9.1 that allows multiple correct choices. The respondent should provide all correct choices, separated by spaces. Provide instructions in the question text.

★ **P9.7** Add a class `AnyCorrectChoiceQuestion` to the question hierarchy of How To 9.1 that allows multiple correct choices. The respondent should provide any one of the correct choices. The answer string should contain all of the correct choices, separated by spaces.

★ **P9.8** Add a method `addText` to the `Question` class of How To 9.1 and provide a different implementation of `ChoiceQuestion` that calls `addText` rather than storing an array list of choices.

★★ **P9.9** Provide `toString` and `equals` methods for the `Question` and `ChoiceQuestion` classes of How To 9.1.

★ **P9.10** Implement a subclass `Square` that extends the `Rectangle` class. In the constructor, accept the x- and y-positions of the *center* and the side length of the square. Call the `setLocation` and `setSize` methods of the `Rectangle` class. Look up these methods in the documentation for the `Rectangle` class. Also supply a method `getArea` that computes and returns the area of the square. Write a sample program that asks for the center and side length, then prints out the square (using the `toString` method that you inherit from `Rectangle`) and the area of the square.

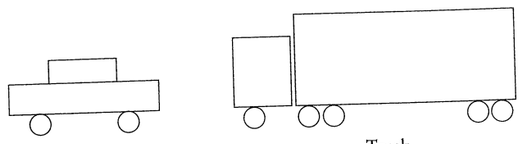
★ **P9.11** Implement a superclass `Person`. Make two classes, `Student` and `Instructor`, that inherit from `Person`. A person has a name and a year of birth. A student has a major, and an instructor has a salary. Write the class declarations, the constructors, and the methods `toString` for all classes. Supply a test program that tests these classes and methods.

★★ **P9.12** Make a class `Employee` with a name and salary. Make a class `Manager` inherit from `Employee`. Add an instance variable, named `department`, of type `String`. Supply a method `toString` that prints the manager's name, department, and salary. Make a

class `Executive` inherit from `Manager`. Supply appropriate `toString` methods for all classes. Supply a test program that tests these classes and methods.

★★★ **P9.13** Reorganize the bank account classes as follows. In the `BankAccount` class, introduce an abstract method `endOfMonth` with no implementation. Rename the `addInterest` and `deductFees` methods into `endOfMonth` in the subclasses. Which classes are now abstract and which are concrete? Write a static method `void test(BankAccount account)` that makes five transactions and then calls `endOfMonth`. Test it with instances of all concrete account classes.

★★★G **P9.14** Implement an abstract class `Vehicle` and concrete subclasses `Car` and `Truck`. A vehicle has a position on the screen. Write methods `draw` that draw cars and trucks as follows:



Car Truck

Then write a method `randomVehicle` that randomly generates `Vehicle` references, with an equal probability for constructing cars and trucks, with random positions. Call it 10 times and draw all of them.

★★G **P9.15** Write a program that prompts the user for an integer, using a `JOptionPane`, and then draws as many rectangles at random positions in a component as the user requested. Use inheritance for your frame class.

★★G **P9.16** Write a program that asks the user to enter an integer n into a `JOptionPane`, and then draws an n-by-n grid. Use inheritance for the frame class.

Programming Projects

Project 9.1 Your task is to program robots with varying behaviors. The robots try to escape a maze, such as the following:

```
* *******
*     * *
* ***** *
* * *   *
* * *** *
*   *   *
*** * * *
*     * *
******* *
```

A robot has a position and a method `void move(Maze m)` that modifies the position. Provide a common superclass `Robot` whose `move` method does nothing. Provide subclasses `RandomRobot`, `RightHandRuleRobot`, and `MemoryRobot`. Each of these robots has a different strategy for escaping. The `RandomRobot` simply makes random moves. The `RightHandRuleRobot` moves around the maze so that it's right hand always touches a wall. The `MemoryRobot` remembers all positions that it has previously occupied and never goes back to a position that it knows to be a dead end.

Project 9.2 Implement the `toString`, `equals`, and `clone` methods for all subclasses of the `BankAccount` class, as well as the `Bank` class of Chapter 6. Write unit tests that verify that your methods work correctly. Be sure to test a `Bank` that holds objects from a mixture of account classes.

Answers to Self-Check Questions

1. To express the common behavior of text fields and text components.
2. Not all bank accounts earn interest.
3. Two instance variables: `balance` and `interestRate`.
4. `deposit`, `withdraw`, `getBalance`, and `addInterest`.
5. `Manager` is the subclass; `Employee` is the superclass.
6. The `SavingsAccount` class inherits the `deposit`, `withdraw`, and `getBalance` methods. The `addInterest` method is new. No methods override superclass methods.
7. It needs to reduce the balance, and it cannot access the `balance` instance variable directly.
8. So that the count can reflect the number of transactions for the following month.
9. It was content to use the superclass constructor without parameters, which sets the balance to zero.
10. No—this is a requirement only for constructors. For example, the `CheckingAccount.deposit` method first increments the transaction count, then calls the superclass method.
11. We want to use the method for all kinds of bank accounts. Had we used a parameter of type `SavingsAccount`, we couldn't have called the method with a `CheckingAccount` object.
12. We cannot invoke the `deposit` method on a variable of type `Object`.
13. The object is an instance of `BankAccount` or one of its subclasses.
14. The balance of `a` is unchanged (you withdraw from and deposit to the same account), and the transaction count is incremented twice.
15. It certainly should—unless, of course, `x` is `null`.
16. If `toString` returns a string that describes all instance variables, you can simply call `toString` on the implicit and explicit parameters, and compare the results. However, comparing the instance variables is more efficient than converting them into strings.
17. Three: `InvestmentFrameViewer`, `InvestmentFrame`, and `BankAccount`.
18. The `InvestmentFrame` constructor adds the panel to *itself*.