```
// Fig. 6.9: PackageDataTest.java
// Classes in the same package (i.e., the same directory)
// can use package access data of other classes in the
// same package.

public class PackageDataTest {
  public static void main( String args[] ) {
    PackageData d;

    d = new PackageData();

    System.out.println( "After instantiation: " );
    System.out.println( "   " + d.toString() );

    d.x = 77;
    d.s = "Good bye";
    System.out.println( "After changing values: " );
    System.out.println( "   " + d.toString() );
  }
}

class PackageData {
  int x;     // Package access instance variable
  String s;  // Package access instance variable

  // constructor
  public PackageData()
  {
    x = 0;
    s = "Hello";
  }

  public String toString()
  {
    return "x: " + x + "    s: " + s;
  }
}
```

```
// Fig. 6.8: Date.java
// Declaration of the Date class.

public class Date {
  private int month;  // 1-12
  private int day;    // 1-31 based on month
  private int year;   // any year

  // Constructor: Confirm proper value for month;
  // call method function checkDay to confirm proper
  // value for day.
  public Date( int mn, int dy, int yr )
  {
    if ( mn > 0 && mn <= 12 )       // validate the month
      month = mn;
    else {
      month = 1;
      System.out.println( "Month " + mn +
                          " invalid. Set to month 1." );
    }

    year = yr;                      // could also check
    day = checkDay( dy );           // validate the day

    System.out.println(
                       "Date object constructor for date " + toSt
ring() );
  }

  // Utility method to confirm proper day value
  // based on month and year.
  private int checkDay( int testDay )
  {
    int daysPerMonth[] = { 0, 31, 28, 31, 30,
                           31, 30, 31, 31, 30,
                           31, 30, 31 };

    if ( testDay > 0 && testDay <= daysPerMonth[ month ] )
      return testDay;

    if ( month == 2 &&   // February: Check for leap year
         testDay == 29 &&
         ( year % 400 == 0 ||
           ( year % 4 == 0 && year % 100 != 0 ) ) )
      return testDay;

    System.out.println( "Day " + testDay +
                        " invalid. Set to day 1." );
```

```
    return 1;  // leave object in consistent state
  }

  // Create a String of the form month/day/year
  public String toString()
  { return month + "/" + day + "/" + year; }
}
```

```java
// Fig. 6.8: Employee.java
// Declaration of the Employee class.

public class Employee {
  private String firstName;
  private String lastName;
  private Date birthDate;
  private Date hireDate;

  public Employee( String fName, String lName,
                   int bMonth, int bDay, int bYear,
                   int hMonth, int hDay, int hYear)
  {
    firstName = fName;
    lastName = lName;
    birthDate = new Date( bMonth, bDay, bYear );
    hireDate = new Date( hMonth, hDay, hYear );
  }

  public String toString()
  {
    return lastName + ", " + firstName +
      "  Hired: " + hireDate.toString() +
      "  Birthday: " + birthDate.toString();
  }
}
```

```
// Fig. 6.8: EmployeeTest.java (modified for Application)
// Demonstrating an object with a member object.

public class EmployeeTest {
  public static void main( String args[] ) {
    Employee e;

    e = new Employee( "Bob", "Jones", 7, 24, 49,
                      3, 12, 88 );

    System.out.println( e.toString() );
  }
}
```

```
// Time2 class definition
import java.text.DecimalFormat;       // used for number formatt
ing

public class Time3 {
  private int hour;     // 0 - 23
  private int minute;   // 0 - 59
  private int second;   // 0 - 59

  // Time3 constructor initializes each instance variable
  // to zero. Ensures that each Time3 object starts in a
  // consistent state.
  public Time3() { setTime( 0, 0, 0 ); }

  // Time3 constructor: hour supplied, minute and second
  // defaulted to zero
  public Time3( int h ) { setTime( h, 0, 0 ); }

  // Time3 constructor: hour and minute supplied, second
  // defaulted to zero
  public Time3( int h, int m ) { setTime( h, m, 0 ); }

  // Time3 constructor: hour, minute, and second supplied
  public Time3( int h, int m, int s ) { setTime( h, m, s ); }

  // Set Methods
  // Set a new time value using military time. Perform
  // validity checks on the data. Set invalid values
  // to zero.
  public Time3 setTime( int h, int m, int s )
  {
    hour = ( ( h >= 0 && h < 24 ) ? h : 0 );
    minute = ( ( m >= 0 && m < 60 ) ? m : 0 );
    second = ( ( s >= 0 && s < 60 ) ? s : 0 );

    return this;  // enables chaining
  }

  // set the hour
  public Time3 setHour( int h ) {
    hour = ( ( h >= 0 && h < 24 ) ? h : 0 );

    return this;  // enables chaining
  }

  public Time3 setMinute( int m ) {
    minute = ( ( m >= 0 && m < 60 ) ? m : 0 );
```

```
    return this;  // enables chaining
  }

  public Time3 setSecond( int s ) {
    second = ( ( s >= 0 && s < 60 ) ? s : 0 );

    return this;  // enables chaining
  }


  // Get Methods
  // get the hour
  public int getHour() { return hour; }

  // get the minute
  public int getMinute() { return minute; }

  // get the second
  public int getSecond() { return second; }

  // Convert time to String in military-time format
  public String toMilitaryString()
  {
    DecimalFormat twoDigits = new DecimalFormat( "00" );

    return twoDigits.format( hour ) +
      twoDigits.format( minute );
  }

  // Convert time to String in standard-time format
  public String toString()
  {
    DecimalFormat twoDigits = new DecimalFormat( "00" );

    return ( ( hour == 12 || hour == 0 ) ? 12 : hour % 12 ) +
      ":" + twoDigits.format( minute ) +
      ":" + twoDigits.format( second ) +
      ( hour < 12 ? " AM" : " PM" );
  }

  // Increment the time by one second
  public void inc() {
    second = (second + 1) % 60;

    if (second == 0) {
      minute = (minute + 1) % 60;

      if (minute == 0) {
```

```
          hour = (hour + 1) % 24;
        }
      }
   }
}
```

```
// Test of Time3 -- method chaining

public class Time3Test {
  public static void main( String args[] ) {
    Time3 t = new Time3();

    t.setHour( 18 ).setMinute( 30 ).setSecond( 22 );
    System.out.println( "Military time: " +
                        t.toMilitaryString() );
    System.out.println( "Standard time: " +
                        t.toString() );

    System.out.println( "New standard time: " +
                        t.setTime( 20, 20, 20 ).toString() );
  }
}
```

```java
// Fig. 7.4: Point.java
// Definition of class Point

public class Point {
   protected int x, y; // coordinates of the Point

   // No-argument constructor
   public Point() { setPoint( 0, 0 ); }

   // Constructor
   public Point( int a, int b ) { setPoint( a, b ); }

   // Set x and y coordinates of Point
   public void setPoint( int a, int b )
   {
      x = a;
      y = b;
   }

   // get x coordinate
   public int getX() { return x; }

   // get y coordinate
   public int getY() { return y; }

   // convert the point into a String representation
   public String toString()
      { return "[" + x + ", " + y + "]"; }
}
```

```java
// Fig. 7.4: Circle.java
// Definition of class Circle

public class Circle extends Point {  // inherits from Point
   protected double radius;

   // No-argument constructor
   public Circle()
   {
      // implicit call to superclass constructor occurs here
      setRadius( 0 );
   }

   // Constructor
   public Circle( double r, int a, int b )
   {
      super( a, b );  // explicit call to superclass constructor
      setRadius( r );
   }

   // Set radius of Circle
   public void setRadius( double r )
      { radius = ( r >= 0.0 ? r : 0.0 ); }

   // Get radius of Circle
   public double getRadius() { return radius; }

   // Calculate area of Circle
   public double area() { return Math.PI * radius * radius; }

   // convert the Circle to a String
   public String toString()
   {
      return "Center = " + "[" + x + ", " + y + "]" +
             "; Radius = " + radius;
   }
}
```

```
// Fig. 7.4: Test.java (modified for application)
// Casting superclass references to subclass references

import java.text.DecimalFormat;

public class Test {
  public static void main( String args[] ) {
    Point pointRef, p;
    Circle circleRef, c;

    p = new Point( 30, 50 );
    c = new Circle( 2.7, 120, 89 );

    System.out.println( "Point p: " + p.toString() );
    System.out.println( "Circle c: " + c.toString() );

    DecimalFormat precision2 = new DecimalFormat( "#.00" );

    // Attempt to treat a Circle as a Point
    pointRef = c;   // assign Circle to pointRef
    System.out.println( "Circle c (via pointRef): " +
                  pointRef.toString() );;

    // Treat a Circle as a Circle (with some casting)
    circleRef = (Circle) pointRef;  // cast super to sub
    System.out.println( "Circle c (via circleRef): " +
                  circleRef.toString() );
    System.out.println( "Area of c (via circleRef): " +
                  precision2.format( circleRef.area() ) );

    // Attempt to refer to Point object
    // with Circle reference
    circleRef = (Circle) p;    // line 33 in Test.java
  }
}
```

```
// Fig. 7.9: Employee.java
// Abstract base class Employee

public abstract class Employee {
   private String firstName;
   private String lastName;

   // Constructor
   public Employee( String first, String last )
   {
      firstName = first;
      lastName = last;
   }

   // Return the first name
   public String getFirstName() { return firstName; }

   // Return the last name
   public String getLastName() { return lastName; }

   public String toString()
      { return firstName + ' ' + lastName; }

   // Abstract method that must be implemented for each
   // derived class of Employee from which objects
   // are instantiated.
   abstract double earnings();
}
```

```
// Fig. 7.9: Boss.java
// Boss class derived from Employee

public final class Boss extends Employee {
   private double weeklySalary;

   // Constructor for class Boss
   public Boss( String first, String last, double s)
   {
      super( first, last );  // call superclass constructor
      setWeeklySalary( s );
   }

   // Set the Boss's salary
   public void setWeeklySalary( double s )
      { weeklySalary = ( s > 0 ? s : 0 ); }

   // Get the Boss's pay
   public double earnings() { return weeklySalary; }

   // Print the Boss's name
   public String toString()
   {
      return "Boss: " + super.toString();
   }
}
```

```
// Fig. 7.9: CommissionWorker.java
// CommissionWorker class derived from Employee

public final class CommissionWorker extends Employee {
   private double salary;      // base salary per week
   private double commission;  // amount per item sold
   private int quantity;       // total items sold for week

   // Constructor for class CommissionWorker
   public CommissionWorker( String first, String last,
                            double s, double c, int q)
   {
      super( first, last );  // call superclass constructor
      setSalary( s );
      setCommission( c );
      setQuantity( q );
   }

   // Set CommissionWorker’s weekly base salary
   public void setSalary( double s )
      { salary = ( s > 0 ? s : 0 ); }

   // Set CommissionWorker’s commission
   public void setCommission( double c )
      { commission = ( c > 0 ? c : 0 ); }

   // Set CommissionWorker’s quantity sold
   public void setQuantity( int q )
      { quantity = ( q > 0 ? q : 0 ); }

   // Determine CommissionWorker’s earnings
   public double earnings()
      { return salary + commission * quantity; }

   // Print the CommissionWorker’s name
   public String toString()
   {
      return "Commission worker: " + super.toString();
   }
}
```

```
// Fig. 7.9: PieceWorker.java
// PieceWorker class derived from Employee

public final class PieceWorker extends Employee {
   private double wagePerPiece; // wage per piece output
   private int quantity;        // output for week

   // Constructor for class PieceWorker
   public PieceWorker( String first, String last,
                       double w, int q )
   {
      super( first, last );  // call superclass constructor
      setWage( w );
      setQuantity( q );
   }

   // Set the wage
   public void setWage( double w )
      { wagePerPiece = ( w > 0 ? w : 0 ); }

   // Set the number of items output
   public void setQuantity( int q )
      { quantity = ( q > 0 ? q : 0 ); }

   // Determine the PieceWorker’s earnings
   public double earnings()
      { return quantity * wagePerPiece; }

   public String toString()
   {
      return "Piece worker: " + super.toString();
   }
}
```

```java
// Fig. 7.9: HourlyWorker.java
// Definition of class HourlyWorker

public final class HourlyWorker extends Employee {
   private double wage;   // wage per hour
   private double hours;  // hours worked for week

   // Constructor for class HourlyWorker
   public HourlyWorker( String first, String last,
                        double w, double h )
   {
      super( first, last );   // call superclass constructor
      setWage( w );
      setHours( h );
   }

   // Set the wage
   public void setWage( double w )
      { wage = ( w > 0 ? w : 0 ); }

   // Set the hours worked
   public void setHours( double h )
      { hours = ( h >= 0 && h < 168 ? h : 0 ); }

   // Get the HourlyWorker's pay
   public double earnings() { return wage * hours; }

   public String toString()
   {
      return "Hourly worker: " + super.toString();
   }
}
```

```
// Fig. 7.9: Test.java
// Driver for Employee hierarchy
import java.text.DecimalFormat;

public class Test {
  public static void main( String args[] ) {
    Employee ref;  // superclass reference
    Boss b;
    CommissionWorker c;
    PieceWorker p;
    HourlyWorker h;

    DecimalFormat precision2 = new DecimalFormat( "#.00" );

    b = new Boss( "John", "Smith", 800.00 );
    c = new CommissionWorker( "Sue", "Jones",
                              400.0, 3.0, 150);
    p = new PieceWorker( "Bob", "Lewis", 2.5, 200 );
    h = new HourlyWorker( "Karen", "Price", 13.75, 40 );

    ref = b;  // superclass reference to subclass object
    System.out.println( ref.toString() + " earned $" +
                  precision2.format( ref.earnings() ) );
    System.out.println( b.toString() + " earned $" +
                  precision2.format( b.earnings() ) );

    ref = c;  // superclass reference to subclass object
    System.out.println( ref.toString() + " earned $" +
                  precision2.format( ref.earnings() ) );
    System.out.println( c.toString() + " earned $" +
                  precision2.format( c.earnings() ) );

    ref = p;  // superclass reference to subclass object
    System.out.println( ref.toString() + " earned $" +
                  precision2.format( ref.earnings() ) );
    System.out.println( p.toString() + " earned $" +
                  precision2.format( p.earnings() ) );

    ref = h;  // superclass reference to subclass object
    System.out.println( ref.toString() + " earned $" +
                  precision2.format( ref.earnings() ) );
    System.out.println( h.toString() + " earned $" +
                  precision2.format( h.earnings() ) );
  }
}
```